DOCUMENT RESUME

ED 291 359 IR 013 148

TITLE	Extending the Human Mind: Computers in Education. Proceedings of the Annual Summer Conference (5th, Eugene, Oregon, July 31 and August 1, 1986).
INSTITUTION	Oregon Univ., Eugene. Center for Advanced Technology in Education.
PUB DATE	86
NOTE	182p.; For the 1985 and 1987 proceedings, see ED 265 831 and ED 290 433. Some pages contain small, light type.
AVAILABLE FROM	Proceedings, Center for Advanced Technology in Education, 1787 Agate Street, University of Oregon, Eugene, OR 97403 ($15.00 prepaid).
PUB TYPE	Collected Works - Conference Proceedings (021) -- Viewpoints (120) -- Reports - Descriptive (141)
EDRS PRICE	MF01/PC08 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; *Computer Literacy; Computer Software; *Curriculum Enrichment; Databases; Elementary Secondary Education; *Instructional Effectiveness; Keyboarding (Data Entry); *Learning Resources Centers; Liberal Arts; *Microcomputers; Optical Disks; Programing; Robotics; Word Processing

ABSTRACT

The 24 papers in these proceedings describe computer technology and its many applications to the educational process. Topics discussed include computer systems in the classroom of the future; how to evaluate a school's computer program; computer use in special education; computer science and the liberal arts; keyboarding issues; making the computer speak the child's language; going online; useful public domain software for educators; advance techniques used in function graphing; impact of optical storage technology on education; print shop applications in the classroom; creating school documents with PageMaker; new databases in the secondary content areas; the CI4 inservice model; the Apple classroom of tomorrow; the future of computers in instruction; computer integrated schooling; new directions for teaching basic programming; ideal uses of the computer in science; the role of school-based computer coordinators as change agents in elementary school programs; robots in elementary classrooms; teachers use of Superpilot to write software; and critical issues in the teaching of writing with a word processor. (CGD)

****
*   Reproductions supplied by EDRS are the best that can be made   *
*   from the original document.   *
****

ED291359

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

5TH ANNUAL
"EXTENDING THE HUMAN MIND: COMPUTERS IN EDUCATION"
SUMMER CONFERENCE

PROCEEDINGS - '86

UNIVERSITY OF OREGON

IR013148

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
David Moursund
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Proceedings

"Extending the Human Mind: Computers in Education"

Fifth Annual Summer Computer Conference

Sponsored by

Center for Advanced Technology in Education

and

Continuation Center

University of Oregon

Held at the

Eugene Conference Center

Eugene, Oregon

July 31 and August 1, 1986

Published by the Center for Advanced Technology in Education

College of Education

University of Oregon

Eugene, Oregon 97403

© University of Oregon 1985

BEST COPY AVAILABLE

Preface

This is the Fifth Summer Conference in this series sponsored by the University of Oregon. Understanding and using wisely the advanced technology now becoming available to us is a challenge worthy of any educator. We at the University of Oregon are pleased to contribute to that understanding and appropriate use.

The conference is sponsored this year by the Center for Advanced Technology in Education, established by the Oregon State Board of Higher Education, and by the University of Oregon Continuation Center.

Primary roles in the organization of this Conference are played by Paul Katz, Conference Manager, and Keith Wetzel, Program Chair.

The Program Committee also consists of Bruce McKinlay, Dick Ricketts, Jack Turner and Sam Miller.

Robert D. Gilberts, Dean
College of Education

About CATE

and the

Continuation Center

The Center for Advanced Technology in Education (CATE) is a consortium of university-based, community and state organizations with an interest in the use of advanced technology in schools. The educational use of computers is a primary focus of the Center's activities.

Opened in the fall of 1983, CATE is the site of both preservice and inservice training for educators and a field-based research and development program investigating a variety of advanced and interactive instructional technologies and their implications.

Members of the CATE Council and the CATE network are widely scattered across Oregon. They include school districts, state colleges, community colleges, extension services and the State Department of Education. Housed in the CATE facility at 1787 Agate is the ERIC (Educational Resources Information Center) clearinghouse on Educational Management, the International Council for Computers in Education (ICCE), the Instructional Technology (IT) Laboratories, and the Career Information System (CIS). There is also a resource library and facilities for classes and workshops on the use of computers in education.

Leadership in fulfilling the University's outreach responsibilities rests with the University of Oregon Continuation Center, composed of the Community Education Program, Summer Session and Continuing Education. A primary goal of the Continuation Center is to promote educational programs which link research and knowledge in the University with interests and resources in the community. The programs, which include courses, workshops, specialized resources and conferences, must be contemporary and significant. Programs are provided at all levels, serving preschoolers to senior researchers, community professionals and non-professionals. The Continuation Center is concerned with broad-based, pervasive educational issues of our time. Few topics fit this concern so clearly as the Annual Summer Conference "Extending the Human Mind: Computers in Education."

For more information about CATE or the Continuation Center, write to Center for Advanced Technology in Education, 1787 Agate, Eugene, Oregon 97403, or the Continuation Center, 333 Oregon Hall, University of Oregon, Eugene, Oregon 97403.

About The Proceedings

The sponsers of the Fifth Annual Summer Conference "Extending the Human Mind: Computers in Education," the Center for Advanced Technology in Education (CATE) and the University of Oregon Continuation Center, are pleased to offer these proceedings.

Conference presenters deserve appreciation for their promptness in meeting deadlines so the proceedings could become a reality. To meet publication deadlines and control printing costs, the papers are presented exactly as they were received.

The authors of these papers were invited to make presentations at this Conference because of their acknowledged scholarly or professional competence in some aspect of computers and education. The findings, views or opinions expressed by those authors are entirely their own and do not necessarily represent those of CATE, the Continuation Center or the University of Oregon.

Additional copies of the proceedings may be obtained by sending $15.00 (prepaid or purchase order) to Proceedings, Center for Advanced Technology in Education, 1787 Agate Street, University of Oregon, Eugene, Oregon 97403. Checks should be made payable to CATE.

Foreward

This year's Conference focuses on "the classroom of tomorrow" and some of the leading edge technologies it may feature, such as CD-ROM, expert systems software and electronic information exchanges. Other featured topics include classroom proven applications such as keyboarding, graphics, robotics, and logo and written composition.

The papers in these proceedings reflect the experiences and findings of designers, researchers and teachers with new technologies.

In reading these papers, one has to be impressed with the variety of ways in which educators have been applying the computer. The papers illustrate substantial advances in some areas and also raise important questions about research, instruction and management in "the classroom of tomorrow". By providing a forum in which the uses of computers in education can be described and reviewed, the Conference and these proceedings can help us all monitor progress in the field. Such a sampling provides us with an overview of a new and dynamic field, and avoids the narrow specialization that has led education down many blind alleys in the past.

Keith Wetzel
Program Chair
"Extending the Human Mind: Computers in Education" 1986

Table of Contents

1. *Expert Systems in the Classroom of the Future* 1
John Arch
Associate Professor of Computer Science, Shippensburg University

2. *How to Evaluate a School's Computer Program* 8
Karen Billings
Director of Sales, U.S. Education Market, Computer Logo Systems, Inc.

3. *Piloting Computer Use in Special Education* 14
Rosemary Brown
Project EXIT Coordinator, Child Services Center

4. *Computer Science and Liberal Arts* 21
J. Michael Dunlap
Willamette University

5. *Keyboarding: The Issues Today* 33
Ann Fidanque, Teacher, 4J School District
Mary Smith, Teacher, Bethel School District #52
Gail Sullivan, Teacher On Special Assignment - Computers, School District 4J

6. *Making the Computer Speak the Child's Language* 40
Larry Flick
Assistant Professor, Teacher Education, University of Oregon

7. *Going Online: Dialing into the Future* 48
Deborah L. Graham, M.L.S.
Director, Library Services, Sacred Heart Hospital

8. *Useful Public Domain Software for Educators* 53
Kurt Gustafson
Science and Computer Teacher, Greater Albany Public Schools

9. *Advanced Techniques Used in Function Graphing* 58
Dr. Lee Thomas Hill
Professor, Mathematics and Computer Science, Southern Oregon State College

10. *The Impact of Optical Storage Technology on Education* 72
Kathleen M. Hurley
Vice President, Educational Division, Mindscape, Inc.

11. *Applications in the Classroom: Print Shop, Print Shop Companion and Other "Easy" Programs* 75
Brian James
Teacher, Churchill High School, School District 4J

12. *Creating School Documents with PageMaker* 79
Julia A. H. King
Lecturer and Consultant, University of Oregon Continuation Center

13. *New Databases in the Secondary Content Areas* 85
Tom Layton
South Eugene High School, School District 4J

14. *The CI4 Inservice Model* 91
Judi Mathis, Doctoral Student, University of Oregon
Jim McCauley, Doctoral Student, University of Oregon
Seymour Hanfling, Doctoral Student, Unviersity of Oregon

15. *The Apple Classroom of Tomorrow* 101
Dr. Samuel K. Miller
Project Coordinator and Head Teacher, Eugene's Apple Classroom of Tomorrow

16. *The Future of Computers in Instruction* 107
David Moursund
Professor, College of Education and Department of Computer Science
University of Oregon

17. *Computer Integrated Schooling* 122
Michael J. Neill
Director, Indiana Clearinghouse for Computer Education

18. *New Directions for Teaching Basic Programming* 129
Andrew M. Paulson
Math and Computer Teacher, Lowell High School

19. *Ideal Uses of the Computer in Science* 138
Bill Sherzer
Science and Computer Teacher, Lowell Junior High School

20. *The Role of School-Based Computer Coordinators as Change Agents in Elementary School Programs* 144
Neal Strudler
Graduate Teaching Fellow, College of Education, University of Oregon

21. *At First You Succeed* 151
Gail Sullivan
Teacher On Special Assignment - Computers, School District 4J

22. *Robots Go To Elementary Classrooms!* 155
Elizabeth Anne Viau
Graduate Teaching Fellow, Computers in Education
University of Oregon

23. *Teachers Write Software with Superpilot* 163
Elizabeth Anne Viau
Graduate Teaching Fellow, Computers in Education
University of Oregon

24. *Critical Issues in the Teaching of Writing with a Word Processor* 170
Keith Wetzel
University of Oregon

EXPERT SYSTEMS IN THE CLASSROOM OF THE FUTURE

by

John Arch
Associate Professor of Computer Science
Shippensburg University
Shippensburg, PA 17256

Summer 1986

What is an Expert System?

An expert may be defined as a professional's professional; one of the best practitioners in a particular field. By this definition not every professional, be it a physician, chemist, or teacher, is an expert. An expert system is a computer program that is designed to solve problems at the same level of competence as a human expert. Theoretically, if human expertise can be defined for a particular field, an expert system should be possible to design for that field. Construction of an expert system is a complex undertaking, often growing into a project involving thirty or more person years of development.

To date expert systems have been developed only in areas where human expertise can be clearly defined. Technical fields have offered the most promise because these areas have the most exactly defined 'right' answers. Some examples of expert systems currently in varying stages of development include:

DENDRAL - to predict the chemical structure of organic compounds given the mass spectograph of an unknown compound;

INTERNIST - to help diagnose diseases in the field of internal medicine

MYCIN - to help physicians diagnose and treat certain types of infectious diseases;

PROSPECTOR - to help geologists locate mineral deposits.

While these systems sometimes have equalled or exceeded the expertise of humans, as yet none is widely used outside the research lab.

These examples are indicative of the scientific bent of most of today's expert systems. Human experts seem to reach consensus more readily when there are areas of definite agreement supported by experience and research. Developing an expert system in a non-technical field, such as literary criticism, would bog down over questions of which school of criticism is 'right' or 'more right' in a given situation. In fact, right and wrong are not readily identifiable in many areas of human endeavor, and such fields are not likely candidates for expert systems.

What Has Been Done in Education With Expert Systems?

At present expert systems in education fall into two categories. These are:

Systems to tutor students. Two such systems are

GUIDON -	to help medical students study infectious diseases;
SCHOLAR -	to help students learn the basic facts about the geography of South America.

Systems to help train teachers. An example is

BUGGY -	to help train teachers to determine patterns in students' arithmetic errors. A series of problems is presented containing the same error (or bug). The teacher tries to determine the pattern of the bug. When the teacher believes he or she has figured out what the bug is, several problems are given to see if he or she is correct.

Every educational expert systems contains three main parts: a knowledge module, a learning module, and a tutoring module.

The knowledge module, or knowledge base, consists of the facts, rules, and procedures of the particular subject area covered by the system. It is developed by questioning expert educators and content area experts and by consulting the research. It is against the knowledge base that student responses are compared to determine if they are correct or incorrect.

The learning module consists of a model of how a student learns and is designed to determine why a student has made a particular error. Ideally this module will detect any patterns that may appear in the student's errors.

The tutoring module contains strategies about how to teach a particular skill to a student.

What Obstacles Must be Overcome?

While expert systems promise great benefits in the future there are some difficult problems to solve first.

Defining expertise

Defining human expertise has proved a more difficult problem than expected. Often, even human experts are not certain why they do what they do when they do. Even in fields with strong scientific foundations, such as medicine or science, much of what an expert does is as much art as science.

Developers of expert systems must conduct detailed interviews with human experts in order to discover a consensus of what the correct course of action should be in a given situation. This process of organizing human expertise so that it can be understood well enough to be programmed into an expert system is called knowledge engineering.

Knowledge engineering is not simply a matter of identifying a set of experts and asking them to tell all they know. Rather, the knowledge base for expert systems is usually built from the bottom up, by asking experts questions such as, 'Given a particular situation, what would you do and why?'. As is apparent, it can take a great deal of time to coordinate the answers and opinions of a number of leading experts in a particular field, particularly when disagreements crop up.

When developing educational expert systems questions that must be answered include: What is it that a teacher does that can be defined as expertise? How does a student learn? How can an expert educator be identified?

Expanding beyond math and science

As mentioned previously, most expert systems deal with technical fields such as math and science. While these areas are important, education has many other facets. If expert systems are going to do all they to improve education they must be expanded beyond the narrow technical fields to which they are now confined.

Overcoming human resistance

Human experts may not take kindly to being 'told' what to do by a computer or, worse yet, being replaced by an expert system. To date most job loss and displacement due to computers has been in blue collar or secretarial areas, but expert systems may eventually pose a real threat to white collar professionals - physicians, chemists, educators, and so on. The hope is that rather than cause humans to rebel, the arrival of expert systems will spur humans to strive for even greater levels of expertise.

What Special Problems Does Education Pose to the Development of Expert Systems?

In addition to the general problems that must be overcome when developing expert systems, educational expert systems pose some special problems.

Problems defining goals

Where should the money be spent in educational expert system research? In other words, what should educational expert systems be used to do? Three areas seem possible at present. First, expert systems can be developed to actually teach students. Second, they could be used to help train teachers, as is the goal of BUGGY. Finally, systems could be developed to serve as teacher consultants.

Not much energy has been put into this third area. A teacher consultant expert system is a program that contains the accumulated knowledge of a particular branch of the teaching profession and makes it available instantaneously to educators. Thus, a teacher who wanted to know what the research says about a problem teaching skill X to student Y aged Z would be able to get fast up-to-date advice by pressing a few keys on a computer. Consultation programs are already being developed in other fields; witness INTERNIST and PROSPECTOR mentioned earlier. The same type of program can be developed to assist teachers.

Problems with hardware

Microcomputers currently found in schools do not have sufficient memory to develop or run any but the most elementary expert systems. Each of the systems described earlier in this paper was developed on mini- or mainframe computers.

Problems with natural language

Natural language is the everyday language humans speak, and while most people become fairly proficient at it, natural language can cause computers some problems. For example, the sentence 'We saw the fruit fly above the orange' is perfectly clear to people, but might not be to a computer.

If educational expert systems are going to make progress in interacting with students in the students' own language, these problems are going to have to be overcome.

Developing Educational Expert Systems

Choose an appropriate topic

A topic for an educational expert system must be one with clearly definable expertise. One the author is currently involved with is the development of a computerized consultant for elementary reading teachers. Other possible topics for educational expert systems include the natural and biological sciences, grammar, geography, and punctuation.

Interview the experts

Get out into the field and talk to the educational experts, the teachers themselves. There is a great deal of expertise is any school district just waiting to be tapped. The author's experience with reading teachers has shown that while not every reading teacher is an expert there are some who are. Expertise can also be found in the literature.

Implement the system

Most of the expert systems that have been developed have been written in LISP on mini- or mainframe computers. Most teachers do not know LISP, but LOGO is similar and there have been expert systems written in LOGO. Pascal, and perhaps even BASIC, could also be used.

The memory of most school computers (64 or 128K) might not be enough for any but the most elementary of expert systems, but within the next few years this problem should take care of itself, as memories will continue to increase in size.

What Does the Future Hold?

While predictions are easy to make they are often, if not usually, wrong. Nevertheless, given the current level of research in expert systems the following seem likely by the turn of the century.

- Expert systems will be serving as teacher consultants, much as the medical expert systems currently under development are intended to serve physicians.
- Expert systems will be used to train teachers.
- Expert systems will not be widely used in the actual teaching of students except in narrow, clearly defined subject areas. Higher level cognitive processes will become fair game for expert systems only when the obstacles presented by natural language are overcome.
- The appearance of expert systems will force educators to examine their role in the entire educational process. As computers get better teachers will get better.

HOW TO EVALUATE A SCHOOL'S COMPUTER PROGRAM

Karen Billings

The teachers in the Classroom of Tomorrow will implement many new educational programs. New programs in the traditional content areas will come about because of new or revised curricular goals and objectives, new instructional resources, or new teaching methods. New programs will also develop because there is new content to teach, as with the computer literacy or computer science courses.

In some of the Classrooms of Tomorrow, new educational programs will be implemented under the best of conditions: creative and competent teachers, abundant and appropriate resources, effective administrators, and a supportive local community. The programs seem destined to succeed.

In other classrooms or schools, the programs seem instead, destined for failure. They are hastily planned by people and then implemented by teachers who are not adequately trained or supported by their administrators. Most of the time the program staff is asked to work with inadequate resources.

At the end of the first year, it is usually clear to the teachers and the program planners that their program has either been a success or that it has problems. Either way, those who gathered information during the first part of the implemention can have a greater chance of success the next year. Both programs need review and monitoring.

The purpose of this paper is to review the need for evaluating computer education programs in the schools, and then describe an evaluation procedure that can work in the classrooms of today but can result in better programs for the Classrooms of Tomorrow.

The Need for Evaluation

Teachers want to review their computer courses or their computer-related instruction to see what needs to be changed from one semester to the next. Building principals and computer coordinators want to periodically review their program to see if its goals and objectives should be changed, to see what type of teacher/staff training is needed, to find out if the appropriate hardware, software, and print materials are being used, and if students are learning what is expected. District personnel may want to know how the computer program is going so that they can share the information with the community or make a report to the school board. And many program evaluations occur because the funding agency requests them.

In fact, any educational program should include an evaluation phase so that information can be gathered and used in the next stage of planning. An evaluation, if carried out well, will help improve the computer education program from year to year. The program staff not only gets information that they can use in a revision of the program, but information that they can convey to interested parties.

The Purpose of the Process

An evaluation process needs both procedures and tools so that educators can gather and then use information about their computer education program. Objectives of an evaluation process are two-fold:

1. to gather information about the major components of the program
2. to use the information to develop conclusions and recommendations regarding the components of the program.

A school's computer education program may be at a very early stage of development. In this case, the evaluation process will help educators react to a program plan or any newly developed goals and objectives. Some programs have been implemented for several years and have the courses, teachers, computer hardware, software, and other instructional materials in place. Then an evaluation process can be used to examine how well these components are working and what, if anything, should be changed.

While any evaluation process has a common set of steps, the process described in this paper attends to the unique characteristics of a computer education program. The process assumes that the computer-related goals and objectives change more often than those in the traditional content areas. And while the public is interested in the results of any educational program, there seems to be more urgency to review the results of computer use.

Planning the Process

In planning any evaluation process, it is helpful to remember that it is an evaluation process and not a research project. If the program is in its early stages, a formative evaluation is probably more appropriate than a summative one. The formative evaluation process allows program staff to gather information for themselves, so that they can review what worked well and what didn't. Evaluation information can help the computer education program move being an experimental one to being a successful one. Then a summative evaluation can be done.

There will be concerns and questions about any evaluation process. Here are just a few of them:

Isn't it too early to evaluate our program? We just started it!
Why do we need to evaluate our program at all? It seems to be going O.K!

Will this information be used on teacher's performance reviews?
Is evaluation worth the effort? Reports just sit on the shelf!
Isn't evaluation just another word for pre-test and post-test scores? To get data, you end up focusing on situations that are easy to measure and unimportant.

Find out the concerns and assumptions from those involved in the evaluation. The concerns need to be dealt with adequately to get the cooperation of everyone. Make sure that everyone understands the purpose of the evaluation and that anyone who contributes to it gets something out of it!

What Will Be Evaluated?

Before data can be collected, specific questions need to be developed about aspects of the computer education program. Rather than ask questions about the program in general, ask questions about certain components: the program goals and objectives, staff development, instructional materials, and student accomplishments. Although this paper will discuss these four main components, you can review any one component, or additional components of the program.

Program goals and objectives. The program goals and objectives change as hardware costs decrease, as the access to computers increases, as new programming languages or software packages become available. Reviewing course or program goals and objectives offers information that helps you decide which objectives should be changed, added, or deleted. Compare the objectives to the school's philosophy, community needs, the access to appropriate instructional materials, or the teacher training.

Staff training. Information needs to be gathered about all the staff in the computer education program, not just the teachers. Constant feedback is needed regarding the preparation of the staff. What has been successful in the past and what is needed in the future?

Instructional materials. School personnel select and use computer hardware, software, and print/non-print materials for use with students. What actually gets used and how satisfactory are those materials? How well do they actually support the objectives of the program and what new materials are needed for next year?

Student accomplishments. Teachers and students want the computer education program to have a positive effect on students. What kinds of opportunities did students actually have while using a computer? What did students accomplish by using the instructional materials that have been provided? What unexpected outcomes occurred as a result of computer use?

What Are the Steps?

Upfront planning does pay off, so after you understand the purpose of the evaluation and know what aspects of the program to evaluate, you will be even more successful in gathering information by reviewing things such as the following:

- Are there any standard or district procedures that must be followed?
- Who is the primary audience for the evaluation - what and how much do these people want to know? What kind of a report do they want?
- What does the evaluation really mean to the school - how will the results be used?

Gathering the Information

After reviewing the evaluation process and having a general sense of the work involved, the information needs to be gathered. The steps in gathering the information are to develop the questions, select or design the tools, and finally to use the tools.

Developing the Questions

Questions need to be formulated about any of the components about which you want information. If your questions are relevant, people will want to help you answer them, and your report will be of interest to them. If your questions are specific, responses are more easily obtained. Here are some examples of questions:

Question: How is the computer education program going?
Better question: Is the program fulfilling its goals or objectives?
Even better questions: Does the program staff think that each goal or objective is being met? If so, how well was it met? If not, why not?
How do the objectives that were actually implemented compare to those that were intended?
Is each program goal/objective attainable with current access to computer hardware, software, or other instructional materials, or given the extent of teacher training?
What program goals/objectives should be added, revised or deleted?

Selecting or Developing the Tools

After developing the evaluation questions you will need to find or develop tools to help answer them. There are various tools from which to choose, including questionaires, observations, interviews, document reviews, and tests. Any tool can help you gather information and respond to a question, but some are easier to administer, while others are more labor-intensive but may give more detailed information.

There are few, if any, available tools yet that are useful for doing a formative evaluation of a computer education program. Your school's program may have unique goals and objectives, for which no test items have been developed. Many computer educators are using questionnaires or checklists, developed years ago for reviewing teacher inservice programs or selecting instructional materials, but are now updating and revising them for use in evaluating the computer program.

Using the Tools

Once the tools are designed, it is simply a matter of scheduling the time periods when they can be administered, and then using the tools. Given the recent development of computer education programs, many people are going to be interested in finding out what really is being taught and how, and therefore in tools that help gather that data. If you are looking at program goals and objectives, teacher training, or the use of instructional materials, you might fill in columns on a chart, to discover which of the things planned were actually implemented.

There are some limitations in the data collection: specific time periods when pre and post tests can be given or when teachers can be interviewed. Do you need to ensure confidentiality of responses to any forms or to request permission to review certain student records?

Using the Information

Just as asking the right questions and developing the right tools are important to the data collection process, summarizing the information and developing recommendations are important to the improvement of the computer education program.

Summarizing the Information

The information has to be summarized well so that the results can be presented well, either in written or verbal form. The more you structured the responses in your instruments, the more easily you can summarize them. For example, responses with a numerical rating scale are already categorized and can easily be tallied and totaled. The responses can then be converted to percentages or graphed to give a quick glimpse of the information.

Making the Recommendations

The information that you get from the evaluation is supposed to be used for program improvement. People interested in the data you collected will also want you to tell them what should happen next.

Formulate your recommendations around the program components, just as you formulated the questions about each one. Within in each of the components, try the following approach:

1. Recommend first those things that don't need to change because they are working well. Share the success stories up front.

2. Recommend next those things that would help reinforce or extend aspects of the program. Reviewing some of the success stories in the program will give you information with which to make these recommendations.
3. Offer suggestions for any revisions or restructure of the program. While it is easier to talk about the success stories than to fix a problem, the fixes need to be reported and discussed.

Reporting the Information

Your final responsibility in the evaluation process is to give a report - verbal or written. Effective reporting of any information adds to its influence; poor reporting of the best of results is likely to make them ineffective. Above all, the report needs to be delivered in time for the decision-makers to study your conclusions and recommendations and then use them in any later decisions.

The report will be shaped by the evaluation design, the extent of the computer education program, and the questions about the different components, and, of course, by the audience of the report. Make it clear where the important information is located so the report can be understood quickly and easily. This ensures that more people will use the results and that your work will have been worthwhile.

References

Billings, Karen J. An Evaluation Handbook for a Computer Education Program. Eugene, OR: ICCE. 1985. 156 pages.

Fink, Arlene, and Kosecoff, Jacqueline. An Evaluation Primer. Beverly Hills, CA: Sage Publications. 1978. 99 pages.

Morris, Lynn Lyons, and Fitz-Gibbon, Carol Taylor. Evaluator's Handbook. Beverly Hills, CA: Sage Publications. 1978. 138 pages.

Wolf, Richard M. Evaluation in Education. Second Edition. New York: Praeger. 1984. 214 pages.

PILOTING COMPUTER USE IN SPECIAL EDUCATION

Rosemary Brown

Introducing Project EXIT

Project EXIT is a pilot program of School District #1 in Portland, Oregon. It's mandate is to define and demonstrate methods by which to integrate computer use in Special Education resource rooms and self-contained classes across the District. Now entering their third year, Project participants have come to some considered conclusions about the use of microcomputers in special education classes.

Project EXIT was proposed as a three year project to explore the premise that under the right circumstances microcomputers could be a useful tool in the instruction of special education students. At that time in Portland special education students had very little access to any kind of computer use and there was no training for teachers in how to incorporate computer use with special populations. The Project was designed to provide adequate hardware, software and training to determine whether and how computers might be good for use in special education.

The Classroom Component

In the Fall of 1984 Project EXIT began operation with it's first group of teachers. Standard procedure was for the Project to place an Apple computer with printer into the classroom of each participating teacher and to provide each teacher with a budget for software, but five pairs of teachers asked to share this resource and so the available 18 computers were placed with 23 teachers. Support for Project teachers was provided by a full time Project Coordinator and by part time services from a media specialist. In addition to access to a library of software, each teacher was given a budget of $500 for the purchase of software for his/her classroom and counsel in its selection. Participating teachers agreed to attend monthly Project meetings and to provide monthly data on computer use.

The teachers represented the full range of classrooms for mildly handicapped students in the Portland Public Schools. They taught in resource rooms and self-contained classrooms serving students in kindergarten through twelvth grade across the diverse neighborhoods of this urban district.

Most of the teachers in Project EXIT had little experience with instructional computing until their involvement with the Project. Many had no experience with computers at all.

At the end of the first year of the Project those teachers who wanted to continue were asked to submit proposals indicating what benefit a second year of participation might have. Only four of the twenty-three teachers chose not to continue. In the fall of the second year and again mid-year additional equipment became available, the Project expanded to a total of thirty-eight participants, and the model of two teachers sharing a computer was discontinued as unworkable. The third year will begin with a group of forty-seven teachers some of whom are just beginning to use computers and some of whom will be with their third year in the Project. The present mix of more experienced computer users with those just starting has proven to be an efficient way for participants to learn. A teacher who has experience with one aspect of computer use has practical information to share with one who has not yet investigated that area. In the process of explanation both become more aware of the issues and processes involved. Although those with more experience in the Project are generally more familiar with the possibilities for the instructional use of computers, we have stressed throughout the existence of the Project that there are no "Experts" and that we best serve ourselves and our students when we all share the roles of learner and teacher. In short, none of us knows as much as all of us.

Although a clearly defined project, with restricted participation and a matrix of goals and objectives, Project EXIT functions in a district and a department in which others are also involved in exploring the use of microcomputers. We have made specific efforts to network with other computer users in the Portland School District, including other special education teachers, those working in the mainstream, and those who work with special populations in the English as a Second Language / Bilingual Program, and the Chapter 1 Program. Some of our most valuable exchange has been with teachers of seriously handicapped students. The shared interest in instructional computing has led to the formation of a Special Instruction Special Interest Group (SI-SIG) as an affiliate of the District's Computer User's Group. SI-SIG meetings have attracted teachers, administrators, psychologists, speech / language pathologists, occupational therapists, teacher's aides, and others for presentations on ranging from "My Favorite Software" to a first peek at interactive video.

Computer Use Outside of the Classroom

While the major emphasis of the Project is its classroom component, there are also goals for administrative computer use and for the use of computers by the departments itinerent staff of assessment team personnel. The effect of computer use outside the classroom is to legitimize and build support by the whole department for the importance of computer access by special education teachers and special education students. A speech / language pathologist who has learned to use the AppleWorks data base to track student progress can show that process to a teacher. An administrator who has become accustomed to using a word processor for his/her own writing is not likely to question its value for students. When EXIT was first proposed there were no computers in use in Special Education offices. One major expenditure of the first year was the procurement of six IBM PCs for secretarial and administrative use. The use of these machines was facilitated by District classes in Wordstar and by Project support for secretaries on an on call basis. During the second year administrators attended an overview of software application for management purposes and secretaries attended weekly users' groups in Wordstar and Lotus 123. These activities were very favorably received and will be continued.

Similarily, before Project EXIT there was no computer access for psychologists, social workers, speech / language pathologists, occupational therapists, physical therapists, and the other specialists in the Department. The Project placed two Apple IIe systems in the Staff Development Center and provided training and appropriate software. Psychologists and social workers now use software to quickly and accurately scores tests such as the Vineland, Woodcock-Johnson, WISC-R, and K-ABC. Speech / language pathologists use software for the analysis of certain articulation and language problems. Some of the assessment staff are using the word processing programs to produce evaluation reports, create teaching materials, and track intervention with students. Next year's objectives include training representatives in telecommunicating using SpecialNet and providing continuing access, training, and support for computer use. Assessment staff will be included on the steering committee of the SI-SIG.

Conclusions From the First Two Years

After two years of experience as a Project there are some things that Project participants have found to be true. There is agreement that computers work wonderfully with special education students when teachers have adequate hardware, software, training and support. Even after the initial novelty has worn off computers can be motivating, non-threatening, and instructive. Computers are not intrinsically isolating and can be used as a medium for peer interaction and as an avenue for academic success in a high status format. One teacher insists that practice in being reasonable with the computerhas helped her emotionally disturbed students learn how to be more reasonable with each other. "Oh", she quotes one student,"You mean explain things one at a time like I do with LOGO."

EXIT teachers have found that word processing can break through the defeat and resistance of students who "won't write". Even without instruction in keyboarding students are consistently willing to produce twice as much writing with a word processor and are much more willing to edit and revise their work.

Project teachers are adamant about the fact that computers are terrific tools for special education teachers. They use their Apples for writing assessment and progress reports, for planning lessons, for creating materials, and for managing and analysing student data. Computer use is becoming so routine that one teacher explained that he had come to rely on the information management capabilities it offered him and that he "couldn't teach" without it.

Project teachers have been tracking both the amount of time that their computer is used each month and the way that their computer is used by their students and by themselves. The amount of time spent on various applications will yield information as to the direction of computer use in the Project as a whole and should highlight any difference in computer use based on class type, grade level, or the computer familiarity of the teacher.

It is also clear that teachers new to computers need consultation in order to use them well. In the beginning they need help on how to organize computer space, schedule, rules so that the arrival of the computer doesn't disrupt the existing program. They also need help in figuring out what area of computer use to explore first. Ongoing personal contact is the best way to help teachers use computers well. In Project EXIT this contact is available from the Project Coordinator. In addition each new teacher is assigned a more experienced Project teacher as a "computer buddy". One of the reasons that Project EXIT computers get used and get used well is that there is always someone that teachers can call when they are faced with a machine that won't work, a program that won't print out, or a general state of confusion about what to do next.

We have found that the need for ingormation and support continues after the initial installation period. Ongoing support guarantees that teachers will improve their use of computers for instruction. The information presented during monthly Project EXIT meetings varies according to the needs and interests of Project participants. Technical tips may explain how to print this software with that printer. Instructional ideas may include an informal presentation of one teacher's experience in using a spell checker to reinforce a spelling list or another teacher's use of a branching story making program with his/her language arts class. Project EXIT meetings are held after school and last until five p.m. They have an unprecedented record of attendance.

The Project's emphasis on the importance of networking has led to a rich environment at Project meetings for the sharing of information and ideas. They discuss new software they have used, review computer classes they have taken, and talk about conferences they have attended. Once they have gotten to know each other Project teachers tell stories of computer related success and stories of computer assisted disaster with equal enthusiasm.

Continuing Concerns

Finding good quality software which fits the classroom's learning objectives is a major problem, especially for beginning Project teachers. Teachers want to know what software to select to teach spelling, problem solving, basic math facts, beginning consonent sounds, vocational skills, and all the other facets of the District curriculum. A consultant can sometimes suggest titles to preview but this is too haphazard approach to such a serious issue. All last year a committee of Project teachers met to discuss the problem of locating good quality software for particular applications and audiences. They decided to create a data base of software reviews by Project teachers. A second committee organized an informal software demonstration session for the October teacher inservice day. This activity was very well received and will be repeated next year on a different day with a provision for release time.

Hardware hassles are another ongoing area of irritation for Project teachers. Computers seem fragile and unpredictable to the new user. More that one Project teacher has been chagrined to find that the monitor which "seems to be broken" only needs the contrast adjusted. At other times teachers, sure that they are doing something wrong have wasted hours struggling with a computer which really is broken. Teachers need to know enough about computer hardware so that they don't break anything. Basic hardware and software maintenence is the focus of the first Project meeting each year and computer buddies are encouraged to offer support on hardware questions when new teachers set up their computers. They also need to be able to determine when the computer or the software isn't working. They are encouraged to call as soon as trouble shooting stops being fun because a teacher's time is too important to waste on mechanical malfunctions. With experience the panic and frustration over equipment problems is much diminished and teachers are able to diagnose and fix the problem, call the repair shop, or call for help quite comfortably. Understanding how the machinery works seems generally empowering and interested teachers are referred to appropriate classes in the community.

Even experienced Project teachers continue to struggle with "user hostile" hardware manuals, printers that won't print from certain software, and interface cards that don't interface. Compatability problems and training needs are compounded by the fact that Project hardware includes three types of printers and three interface cards. Experience proves that it is often a ridiculously specific problem which defeats teachers in their efforts to use computers to teach. When a teacher can't print the Logo picture which has been the focus of a student's effort for weeks, it is not an inconsequential matter. There will be one Project meeting devoted exclusively to printers this year. We have also developed a set of instruction sheets on very specific problems such as "How to Print from Delta Drawing on your Okidata Printer", "How to Print Animation Station by Using the Screen Dump on the Uniprint Card", and "How to Print Logo Pictures Using the Logo Toolkit".

A problem which is more subtle and more critical than those above is the fact that teachers can get stuck in a routine of using the computer in ways which are familiar but not necessarily very effective. Addressing this issue requires a greater level of consultant skill than advising on software or untangling software / hardware problems. Next year teachers in their second or third year with the Project will pursue individual computer learning goals. In addition all Project teachers will be encouraged to take classes, attend conferences, and participate in SI-SIG meetings.

Presentations by peers, structured small group discussions, and informal sharing will continue to be features of Project meetings. At the outset of the Project there was an effort to establish the expectation that we would all be experimenting with computer use, sharing what we discovered and re-evaluating our use on the basis of experience. Because it was understood that the experience and opinion of each teacher was valuable and that each teacher was the best judge of what was right for his/her classroom, presentations by peers were seen as opportunities to learn something new rather than as standards for performance or issues for debate.

Summary

Project EXIT is a model program designed to investigate the use of microcomputers in Special Education resource rooms and self-contained classrooms across the district. The Project has developed a process for training teachers in computer use, for defining and locating software to fit instructional objectives, for managing classroom information on computer, and for offering timely and useful support.

Special education administrators will find that Project E.X.I.T. is an affordable, replicable, and sensible model by for the integration of microcomputers in special education programs. Critical to the success of the program has been the availability of a full time Project Coordinator to provide information, training, and consultation with individual teachers in their classrooms.

Because of adequate support by the District Project EXIT computers are used often and well. We have come a long way since we tentatively unpacked our first Apple computers. In writing this account of the Project it has been satisfying to reflect on our progress so far, but even more satisfying to consider plans for the future. We are just beginning .

Computer Science and Liberal Arts

J. Michael Dunlap
Willamette University

Many disciplines in the Liberal Arts have long histories, some going back into antiquity. It is relatively rare, however, to observe the birth, growth, development and maturity of a new discipline within the space of a just a few years. This is particularly rare in the past hundred years. Nevertheless, Computer Science, in little more than a decade, has grown to assert a major influence on the Liberal Arts College and become recognized as a standard part of the curriculum.

Because Computer Science is so new, it has recently undergone the struggle common to all academic disciplines, that of gaining acceptance within the college curriculum. The difficulties experienced by computer science were exacerbated by its rapid growth and its potential for vocational preparation. In addition, academic leaders hare raised questions about the nature of the discipline, its logical fit into the Liberal Arts, and its relationship to "traditional" disciplines. The most important question raised asks whether Computer Science has a body of theory which distinguishes it as a true member of the academic community.

Concerns about the inclusion of Computer Science in the Liberal Arts stem from historical, political and philosophical factors. The historical precedent of not ever having this discipline in the curriculum caused some to believe that the discipline should not be admitted to the liberal arts program. The political problems were mostly related to availability of faculty, equipment and other resources. These problems have been solved over the last decade in many institutions. Philosophical resistance to Computer Science, however, penetrates deeply into the structure of the modern liberal arts college. The arguments over the inclusion of computer science reveal some fundamental conflicts in the definition of the "liberal arts."

Liberal arts colleges suffer from an inability to clearly define their roles and to convince other colleges to hold to common themes. This produces many definitions, and many interpretations, of "liberal arts," each of which operate well on a locally but cannot be applied elsewhere.

The problem is so pervasive that a task force of the Computer Science Accrediting Board was formed to help provide definitions for Liberal Arts Colleges, the Liberal Arts, and to make recommendations about accrediting standards in institutions calling themselves "liberal arts." Even though the members of the task force were all senior faculty or high level administrators in a representative sample of these institutions, they were unable to agree on a common definition for a Liberal Arts College, or even a definition of the Liberal Arts.

In addition to the preceding difficulties, computer science was also required to justify its existence by demonstrating a recognizable body of theory and demonstration of its role in supporting the concepts encompassed by the "liberal arts." The context in which this took place was one of confusion, lack of standard definitions for "liberal arts," and a deep suspicion that computer science was whatever liberal arts was not.

Today, not only has computer science become accepted in most liberal arts colleges, but it is increasingly recognized as a central and beneficial part of the liberal arts curriculum. It is viewed as highly supportive to the goals and objectives of this curriculum, even though the actual definitions of liberal arts are still vague.

In this paper, definitions of the liberal arts will be considered, and will be compared to the content and practice of computer science. Specifically the body of theory which comprises computer science will be examined, and the role of computer science in the liberal arts college explored.

The Liberal Arts

Some scholars contend that liberal arts involves only those studies which have little potential for direct application to the world of work. In the article "Liberal Arts in an age of technology," Miles states that the "Liberal arts are simply the humanities, social studies, and also the fine arts insofar as they are pursued for mental enrichment rather than for vocational purposes. ... literature, language, philosophy, political science, history, sociology, and certainly art, music, and theatre, insofar as we study those fields for appreciation or for historical insight." (1)

Clearly missing from this list are mathematics, all of science, psychology and computer science. The definition does, however have a strong basis in American higher education. It appears that the college programs around the time of the revolution were designed to prevent those of lower class from attaining professional status in the community. As such, undergraduate education was designed to deal with matters which were largely unimportant vocationally and concentrated on improving the quality of life for the learner. Obviously, a part of this study was directly supportive of graduate study and contained considerable amounts of mathematics, language, and science.

Due to the pragmatic realities of funding the modern college, this rather narrow definition meets vigorous resistance today. The current perceptions of parents and educators are that some vocational preparation must be provided at the undergraduate level to meet the needs of society and to justify the high cost of post-secondary education. This current view is very different from the elitist view of 200 years ago.

As a result of increasing opportunity for higher education and and the american ideal of "education for the masses," some scholars have come to view the liberal arts as a cluster of skills and concepts rather than the study of certain limited fields. In addition to useful skills and concepts, many scholars believe that the liberal arts should enrich the mind of the student, ennoble the character, expand horizons, and Increase insight independently of what we do and are vocationally. (2) They often feel that the liberal arts should apply to us as individuals, regardless of whether we are nurses, engineers, business executives, or whatever. (3)

Part of the confusion arises from the changes in undergraduate education from the colonial period to the present, and another part arises from the long standing and natural differences between undergraduate and graduate programs. In the past, graduate education was necessary for entry to a profession. Currently, this requirement has diminished to the point where a common expectation is that one will be able to enter some professions after only an undergraduate experience. This has led directly to a blurring of the distinction between undergraduate and graduate education.

Computer science becomes involved in the argument at this time because it contains elements which are professional and which can be of value after only modest amounts of training. A concentration on this facet, however, obscures the fact that there is a deep and significant theory of computer science. Unfortunately, the graduate institutions and multiversities were the first to recognize the theoretical structure of computer science and were the first to give it a home. This in turn led to deeper suspicion among liberal arts colleges, because the multiversities were perceived to blur the distinction between graduate and undergraduate education.

This "guilt by association" is underscored by Adams in his discussion of failures in liberal arts education at the University of California, Santa Cruz. He offers the following perception of the role of large universities:

> "...the multiversity had warped the meaning of the liberal curriculum. Under the pressures of research and increasing specialization, undergraduate instruction had come to look more and more like a diluted form of graduate training. Packaged in disciplinary majors, the point of general education at the university level seemed not so much to develop political, historical, and cultural understanding and imagination as it did to produce miniature economists, historians, literary critics, or political scientists." (4)

Clearly the differences between the graduate program and the undergraduate have been reduced in larger schools. Critics of these programs feel there has been an abandonment of the central principles of liberal arts which creates a place for vocational preparation at the undergraduate level and leaves the task of preserving the benefits of the liberal arts to the smaller (often private) school. The elitist side of this argument asserts that:

> "Liberal curricula [in the coming decades] will once again be associated more with private than public institutions, more with the signs of wealth and class than the critical and useful understanding of the realities of contemporary life." (5)

Adams cannot help but to point out these differences as part of the problem and suggests that a larger question must be faced: "The larger and more difficult issue was and remains that there is no single, coherent answer to the question of what a liberal education means in contemporary American society." (6)

Part of the problem is the image of the liberal arts in terms of studies engaged in by gentlemen of wealth and power. The liberal arts has not been able "to push its way completely out of the exclusive and elitist tradition of genteel learning and the suspicion that the knowledge it imparts has more to do with sophistication than with active and equal membership in democratic culture." (7)

The tradition of elitism which surrounds the liberal arts hampers development of disciplines which are more likely to enfranchise rather than to disenfranchise the general population. It is becoming increasingly clear that the computer disseminates knowledge and power in direct conflict with the elitist structures of the liberal arts. While many would not include this elitism as a formal part of the study, the fact remains that this argument underlies much of the vocal argumentation regarding the appropriateness of computer science in the curriculum.

An additional point of view attempts to clarity substantial and positive elements of the liberal arts. A major difference between a liberal subject and one that is not is the degree to which there is an emphasis on cognitive skills, rational analysis, and the tools required to be communicative--clear, expressive, imaginative. A liberal course of study invites contemplation, a look inward, an assessment, even a reassessment of self and society. "Something" went on in the old colleges that helped their graduates to be recognized as possessing a certain style, a moral stance, a reliability in taste and values, a capacity to analyze and even to imagine derived from that amalgam of experiences known as "going to college." It is precisely that "something" that liberal arts colleges nurture. (8)

In an increasingly complex world, the emerging computer scientist needs these capabilities to adequately and effectively face the challenges which will be demanded of them and their discipline. Perhaps the most appropriate definition for the computer scientist as well as many others can be found in Great books of the Western World, Volume One: The great Conversation, The substance of a Liberal Education:

> The aim of liberal education is human excellence, both private and public (for man is a political animal). Its object is the excellence of man as man and man as citizen. It regards man as an end, not as a means; and it regards the

ends of life, and not the means to it. For this reason it is the education of free men. Other types of education or training treat men as means to some other end, or at best concerned with the means of life, with earning a living, and not with its ends. The substance of liberal education appears to consist in the recognition of basic problems, in knowledge of distinctions and interrelations in subject matter, and in the comprehension of ideas.

Liberal education seeks to clarify the basic problems and to understand the way in which one problem bears upon another. It strives for a grasp of the methods by which solutions can be reached and the formulation of standards for testing solutions proposed. The liberally educated man understands, for example, the relation between the problem of immorality of the soul and the problem of the best form of government; he understands that one problem cannot be solved by the same method as the other, and that the test that he will have to bring to bear upon solutions proposed differs from one problem to the other.

The liberally educated man understands, by understanding the distinctions and interrelations of the basic fields of subject matter, the differences and connections between poetry and history, science and philosophy, theoretical and practical science; he understands that the same methods cannot be applied in all these fields; he knows the methods appropriate to each.

The liberally educated man comprehends the ideas that are relevant to the basic problems and that operate in the basic fields of subject matter. He knows what is meant by soul, state, god, beauty and by the other terms that are basic to the discussion of fundamental issues. He has some notion of the insights that these ideas, singly or in combination provide concerning human experience.

The liberally educated man has a mind that can operate well in all fields. He may be a specialist in one field. But he can understand anything important that is said in any field and can see and use the light that it sheds upon his own. The liberally educated man is at home in the world of ideas and in the world of practical affairs in the sense of liking the life he finds about him; but will be at home in that world in the sense that he understands it. He may even derive from his liberal education some conception of the difference between a bad world and a good one and some notion of the ways in which one might be turned into the other.

The method of liberal education is the liberal arts, and the result of liberal education is discipline in those arts. The liberal artist learns to read, write, speak, listen, understand, and think. He learns to recognize, measure, and manipulate matter, quantity, and motion in order to predict, produce and exchange. (9)

The philosophy widely held at Willamette University recognizes both the importance of classical studies and the present need to train young people for lives beyond college.

> In many respects, a liberal arts education can prepare students for the future as well as, if not better than, a technical education. This is not to suggest that they are the antithesis of one another. Neither should they be considered mutually exclusive. Rather, solid grounding in the liberal arts should be a prerequisite to--or taught in combination with--any one of a variety of specific career-oriented majors. .. A liberal arts education basically is not theoretical and denotes studies (language, philosophy, history, literature, abstract sciences) that provide general and wide ranging knowledge that can be applied abstractly to broad areas. In contrast, a technical education usually deals with the direct application of specific skills to a narrow area and often lacks a theoretical or conceptual base. Unlike the former, it does not intend to develop intellectual capacities such as reason, differentiation, and judgement, or the intangible skills like adaptability, resourcefulness, and self-reliance. In short, the liberal arts education has intrinsic value that goes beyond the job. ..., with the tremendous fluctuations in the economy and the technological revolutions that are occurring with machine-gun rapidity, it is almost impossible for anyone to train for a specific life-long career. Most importantly, it equips students with the mental basics and self-discipline necessary to learn skilled trades, either in a college classroom or in the work place. (10)

Theory of Computer Science

Fields which are a part of the liberal arts are often expected to encompass a body of theory which distinguishes the discipline. A discipline without a strong theoretical underpinning has great difficulty in convincing the classical scholar of its place in the liberal arts.

Computer science has had to face its critics from the liberal arts and has been required to defend the theoretical nature of the discipline. The criticism has been particularly strong because of the obvious connections to vocational training. The difficulty of a suitable defense has been compounded by the enormous appetite of industry for new computing theory pushing the field to a point at which new theory becomes practice almost immediately. Seldom has academia seen this rapid conversion of theory to practice and few models exist elsewhere within the liberal arts to make judgments in this regard.

In addition, theoretical computer science is difficult to define. It is somewhat like a bride, with "something borrowed, something blue, something old and something new." The oldest parts of the field are derived from classical mathematics, the applications are to real world problems, but much of the theory is brand new.

Unfortunately computer science is still not very well defined. It is a field that is changing rapidly and will continue to change for the foreseeable future. It has only recently begun to develop an extensive body of theory and technique which distinguish it as a formal academic discipline. The subfields in computer science are evolving rapidly with new ones being added every year. Some of those discussed below may not exist in 10 years, and others will certainly be added.

One could say that computer science is a collection of sub-fields which deal with information processing and the machines on which the information is processed. Specifically, the theory of such machines, the architecture of the machines, numerical methods used to process the information, languages to describe tasks to the machine, algorithms, data structures for holding information, operating systems which manage the machines, interface of machines and humans, the theory of computation, and artificial intelligence.

Computer programming is a necessary skill for work in computer science. It can honestly be said that computer science begins after one has become a competent programmer. Skill in programming is necessary for advanced work in computer science.

Because computers do not communicate naturally with human beings, it is necessary to establish a common language. Language theory is a branch of computer science which deals with the design and development of computer languages. This subfield includes design of computer languages, construction of compilers, comparisons between languages, improving human to computer communication and translation of human language.

Languages are not independent of the machines on which they run, and a strong relationship exists between computer language and machine organization. Machine organization includes hardware and deals with the design and construction of computing machines. Specialists in this area study various arrangements of computing hardware in an attempt to find machine designs which enhance computation. The subfield makes heavy use of boolean algebra, logic, computer circuits, hardware, and electronics.

Work in machine organization is related to theoretical and practical work on operating systems. The operating system is the master control program of a computer. This program communicates with users, schedules the resources of the system, and records how the machine is used. A considerable amount of research is dedicated to ensuring that a computer system will operate properly under a wide variety of circumstances.

Like operating systems, many branches of computer science depend upon the theory of computation. This area of computer science considers the theoretical and mathematical underpinnings of the field. Work in this area is concerned with the proof of new theorems about machines, operating systems, languages, data arrangements, and methodologies. In addition, it defines computers in abstract ways and then attempts to prove the logical relationships between various abstract machines and existing computer systems. The tools used include mathematics and mathematical proof as well as application of theory to practical situations.

Computer science could not have developed as quickly as it has were it not for careful consideration of programming methods and techniques. This field, called algorithms, is the area of computer science which studies efficiency of both space and time. It studies the time taken for a particular process to be completed on a machine and the efficiency of alternative procedures that solve that same problem. There are many strong relationships between algorithms and theory of computation.

Data structures is related to analysis of algorithms. This is the branch of computer science concerned with forms of information storage such as the relative complexity of information retrieval and how information can be stored for best access. Concepts here include stacks, queues, linked lists, trees, searching and sorting. The work here involves a deep understanding of programming, a quest for efficiency, and a love for mathematics.

As one can see from this discussion, considerable overlap is present between some of the theoretical areas of computer science. This is particularly true of artificial intelligence. AI attempts to program machines to perform tasks which would be considered intelligent if done by a human. One goal of the field is to consider the extent to which machines can simulate human thought. Some of the areas in which AI research is currently conducted include: robotics, natural language translation, understanding human speech, decision making, expert systems, and using data to make inferences.

Some areas of computer science overlap other disciplines. The last sub-field to be considered here is numerical analysis. This area is concerned with the nature of numerical computation and attempts to predict and account for errors. It is also concerned with techniques for computation of problems too large to be performed on existing computers. It should be noted that this field existed before computer science and may well be best classified with mathematics. However, the advent of computers has refreshed the area by providing interesting new problems to study and providing some real application for the knowledge which was previously considered of little practical benefit.

Numerical analysis is a good example of the remarkable overlap which exists between the theory of computer science and the theory of other disciplines. In some cases the interface of two disciplines creates exciting new fields and in others, augments areas of potential research. It is clear that computer science is a very broad field which may well touch nearly every area of information handling by human beings.

Computer Science in the Liberal Arts

The following section addresses the relationship between computer science and the Liberal Arts.

The liberal arts is centered on information processing and computer science provides the theory and knowledge necessary to handle information in automated ways. The long term implication is that computer science will continue to be at the very center of what liberal arts attempts to do. However, in a more significant way computer science is directly related to much of what the liberal arts is attempting to do today, in that it contains the concepts, theory, and practices which develop critical reasoning, language skill, and ability to interpret information.

Many scholars agree that the overarching goal of the liberal arts is the development of an independent intellectual entity. This is achieved through development of abilities in systematic reasoning, skill in planning and organization, ability in problem solving, development of consistent performance, the sense of personal confidence, the ability to find out information, reasoning, and critical analysis.

Each of these objectives are addressed in the liberal arts curriculum and each subject field in the liberal arts attempts to inculcate one or more of the objectives. Computer science, however, is capable of providing instruction in nearly all of the objectives and to offer some unique methods to develop some of these abilities.

There is little question in the mind of experienced people that computer science develops skills in systematic reasoning. The ability to communicate complex ideas to a moron level machine and to achieve accurate results demands systematic and precise reasoning. This reasoning process must be supported by very careful planning and organization. In general, these are considered to be the basic problem solving skills in computer science and serve as the tools upon which much of the remainder of the discipline is based.

In developing consistency of performance, the computer has few peers. If an action is incorrect the computer replies with an error message or simply refuses to complete the task. In no other situation is the human being so often and so completely charged with responsibility for their own consistency of performance. This inturn leads to a richer understanding of the working of the mind and the personal knowledge of individual fallibility which leads ultimately to a growth in personal confidence.

Another requisite skill is the ability to find out information. Experience with the computer helps underscore the importance of documentation and the inadequacies of the same. It places a person in an environment in which the tools of experimentation often yield the only available answers. It teaches a person to read carefully, think carefully, and to determine their own answers to the hardest questions.

The final abilities are in critical reasoning and critical analysis. Computer science deals more completely with the man-machine interface rather than with critical analysis of the works of man. These last two objectives derive basic skills and abilities from computer science but require the humanities to provide the laboratory experiences necessary to develop skills in criticism of human works.

One of the most significant parts of computer science is the laboratory component of nearly all classes. It is in the laboratory that the student applies the knowledge gained in the classroom. First, the laboratory requires that students encounter and successfully deal with a wide variety of different equipment. Most of this equipment is strange and may respond in unexpected ways. Secondly, the student is faced with machines which give instant feedback and require responses. This demands that the student make decisions, implement strategies, and then convey this to the machine. The machine responds again, repeating the cycle.

During interaction with the computer, the student is constantly faced with a rigorous form of problem solving. This activity requires that a hypothesis be formulated, converted into a specific implementation, and finally tested. In most cases the hypothesis is incorrect, and the process must be repeated. Such a sequence of activities provides a unique view of classical problem solving strategies and offers the opportunity to try various problem solving techniques and observe their effectiveness.

In general, the computing laboratory is similar to the science laboratory but the feedback cycles are shortened and the laboratory experience is made more powerful by permitting a wide variety of methodology and procedure.

Computer Science and Other Disciplines

The compatibility of computer science and the liberal arts is demonstrated by more than the development of skills desired in the graduate. Computer science is a partner to many of the theoretical explorations central to disciplines long considered parts of the liberal arts.

If mathematics and the sciences are to be included in the definition of liberal arts, the compatibility can not be questioned. Less well understood are the connections between computer science and Philosophy, English, and Foreign Language.

The study of fundamental machine structures helps provide an environment in which the philosopher can study the effects of determinism and the existence of a totally deterministic environment imbedded in a world which may well not be deterministic at all. In addition, the structural organization of machines and the tools of symbolic logic are highly compatible, providing the philosopher a laboratory in which to explore the implications of logical thought.

English is the study of a language and the literature expressed in that language. Both human and machine languages contain a syntax and a semantics along with specialized forms of style. Computer programs in addition, display a style of expression which is similar to human forms found in literature. While the current literary level of computer programs is rather primitive, the body of literature and the range of ideas expressed in computer programs grows yearly. Some are beginning to recognize that certain uniquely human ideas are better expressed in machine languages.

Development of functional languages in computing go far beyond the narrow confines of Von Neumann languages to develop a yet richer set of ideas. These ideas are essentially nonsequential with respect to time and are therefore similar in structure to some of the highest forms of human language. It is clear that good theoretical work in foreign language may be given a significant spur in its development due to the growth of computer languages, language recognition systems, human speech synthesis and human language translation.

Some of the strongest relationships exist between psychology and computer science. Unfortunately, many consider psychology not to be central to the liberal arts and are only slightly less accepting of computer science.

Regardless of the placement of these two disciplines, the relationships between the field which studies the human mind, and the discipline which studies the thinking machine are dramatic. The relationships include models of the mind, analysis of brain function, visual systems, laboratory experiment and control, "humanizing" computer interfaces, study of information retrieval systems, and artificial intelligence.

Conclusion

Computer science has become an increasingly important part of the liberal arts program over the past decade. It has succeeded in once again raising the issue "What is Liberal Arts" and has demonstrated an ability to teach the fundamental concepts and goals so highly prized in a liberal education. In addition, theory which comprises computer science has been articulated. Finally, it is becoming increasingly clear that computer science is a vital partner to much of the research and study which is essential to maintenance of integrity in the liberal arts.

Notes

(1) Liberal Arts in an age of technology. L. Miles. American Education. 20:2-6 Je '84. Page 2.

(2) Ibid. Page 3.

(3) Ibid. Page 3.

(4) Getting Real: Santa Cruz and the crisis of liberal education. W. Adams Change 16:18-27 My/Je '84. Page 23.

(5) Ibid. Page 26.

(6) Ibid. Page 27.

(7) Ibid. Page 27.

(8) The power of professors: the impact of specialization and professionalism on the curriculum. F. Rudolph. Change 16:12-17 My/Je '84. Page 15.

(9) Great books of the Western World, Robert Maynard Hutchins, Editor in Chief. Volume I. The great Conversation, The substance of a Liberal Education. Encyclopedia Britannica, Inc. Chicago, 1952. Pages 3-5.

(10) The value of a liberal arts education. J. T. Rogers and J. E. Sites. USA Today 113:83-4 ja '85. Page 84.

KEYBOARDING: THE ISSUES TODAY

Ann Fidanque, Mary Smith, Gail Sullivan

Anyone who has spent time in elementary and middle schools, and particularly in a computer lab, realizes that we have made a major leap in understanding the needs of students using computers. A year ago educators were debating the whys of keyboarding for elementary and middle school students. Today we are beyond that preliminary debate and are considering issues which determine the success of a keyboarding program. Addressing these issues, we will include insights we have gained from working directly with students and from current research.

WHY SHOULD KEYBOARDING BE TAUGHT?

Computer expert Alan Kay, speaking at the University of Oregon last Fall, said, "I'm not that interested in computers." That was a surprising statement coming from the inventor of the mouse! He went on, "I'm no more interested in the computer than the musician is interested in the cello. What the musician is interested in, is the music."

His analogy fits the issue of keyboarding. Keyboarding is not a goal in itself; it is not a stand-alone skill. Instead, it is the connecting link that allows the student smooth access to the computer power. The goal is the possibilities for interface with the computer that are opened up to the learner through newly acquired keyboarding skills, not the keyboarding itself.

One of the main reasons for the interest in keyboarding is the recognition that word processing is a most useful application program for students, and one that can be easily integrated into the classroom curriculum. Keyboarding is a necessary preskill for any attempt at word processing. Wetzel (1985) points out the frustration for both the students and teachers when students attempt to use a word processing program without adequate keyboarding skills.

The word processor is an excellent vehicle for teaching keyboarding, and since students will eventually be developing their writing skills within a word processing environment, it is an obvious place to start (Cameron, 1986). A year ago, keyboarding software or short basic programs provided the practice screens for keyboarding lesson; this year, another approach has been evident. More often, teachers are using a word processor to provide the practice screen. This of course necessitates instruction in the use of word processor, but it has proved to be beneficial for both areas, keyboarding and word processing.

The integration of these two skills is an encouraging sign that the real goal of keyboarding is being recognized in classrooms.

WHAT IS KEYBOARDING? HOW IS IT DIFFERENT FROM TYPING?

Keyboarding is the input of alpha-numeric information on electronic equipment using the touch system (Hansen, 1985). Keyboarding is a psychomotor skill. The sense of touch which is employed to learn keyboarding is the physiological basis for acquiring the skill. As with the acquisition of any motor skill, systematic mass practice is necessary.

Keyboarding is the basic interface with the microcomputer; it is far more than an updated version of typewriting. As our principal means for human-computer communication, keyboarding assumes a functional importance far beyond that properly attributed to typewriting (Rhodes, 1985). Typing concerns the output of information. Keyboarding involves the input of information into the computer where it can be changed (edited, manipulated, erased) in a variety of ways as well as the final output. Students' keyboarding needs will be those skills that aid them as they compose at the keyboard, not those of a touch typist who works exclusively on copying material.

CAN YOUNG CHILDREN LEARN TO KEYBOARD?

Surprisingly, business education research has been investigating the age question for over fifty years. A great number of studies have been carried out to determine if young children could learn to touch type on typewriters.

In 1930, the Wood-Freeman Study found that third graders were capable of touch typing on manual typewriters. In 1958, Rowe established that students in the third and fourth grades can indeed master the touch typing technique. Eight and nine-year olds possess necessary fine motor skills, eye-hand coordination and recognition of the alphabet to perform this skill. In 1983, the Cowles study showed that young children can learn to type correctly, and they can do so without frustration. These are just a few of the studies that have verified what music teachers knew all along - finger dexterity and purposeful learning combine to enable students to learn fine motor skills.

The age issue was an academic question until the presence of the computer in the classroom began to have an impact on the curriculum. In 1983 The National Business Educators Association recommended that basic keyboarding skills be taught to elementary school children at the earliest feasible age (Bartholome, 1984). The consensus is that eight to nine years is the best age to introduce keyboarding.

WHEN SHOULD KEYBOARDING BE TAUGHT?

In addition to the consensus of research, there is a common sense answer to another question that brings light to the discussion - What is the goal in learning keyboarding? If the goal is to learn keyboarding as a stepping stone to word processing, then the answer is that keyboarding should be taught immediately before the introduction of word processing (Hedley, 1985). Many schools have opted to teach word processing at fourth grade, immedately preceded by keyboarding instruction.

WHICH KEYBOARDING SKILLS SHOULD BE TAUGHT BEFORE FORMAL KEYBOARDING INSTRUCTION?

To avoid poor keyboarding habits that are so difficult to unlearn, primary students need some specific keyboarding skills. More often these students interact with computers in the early grades, yet they are not expected to use word processing programs, so formal keyboarding instruction is not indicated.

In the Eugene School District 4J keyboarding instruction is recommended at fourth grade. Teachers of earlier grades were concerned about keyboarding - they did not want to create "two finger wonders." In response to these valid concerns, some keyboarding skills have been identified that could and should be practiced by primary grade students who are using computers without the benefit of formal keyboarding instruction (Fidanque, Sullivan, 1985).

GRADE 1: Two hands are used on the keyboard at all times. RETURN key is tapped with the right little finger. The spacebar is tapped with the right thumb.

GRADE 2: Home row keys (ASDF JKL;) are tapped with the proper fingers. Featherlight touch is used at all times.

GRADE 3: Reach up and tap the four vowel keys with the proper fingers (E, I, O, U). Note: the vowel key "A" has already been taught in the home row keys.

Some teachers want to begin word processing prior to fourth grade. In those cases, they should begin a complete keyboarding program at that time.

WHAT ARE THE MAIN CONSIDERATIONS IN A SOUND KEYBOARDING PROGRAM?

Technique, speed, and accuracy are the main considerations in a sound keyboarding program, in that order.

Technique refers to the position of the student, from the feet to the curved fingers. It includes the movement of the fingers and manner of striking the keys. The proper foundation for keyboarding skills begins with technique; it is essentially the same as for touch typing skills taught at the secondary level. Since keyboarding is a cumulative skill, the initial lessons must establish proper technique in order to provide a foundation upon which further skills can be built (Robinson, 1985). Proper technique should be stressed from earliest computer experience. Daily practice should concentrate on the pattern of action of the hands and fingers, rather than on the results of the action, the letters or words on the screen. Mastery of the keyboard is achieved initially through correct posture at the machine, proper techniques for striking keys, accurate fingering of keys, and operation of basic machine parts (Kisner, 1984).

Speed is the second consideration. The goal of keyboarding for elementary students is to keyboard at least as fast as they can handwrite legibly. To establish this individualized goal, a handwriting rate test should be administered before keyboarding instruction is begun. Individualized goals accommodate the range of ability teachers find in any given group of students. The individualized goal benefits students because it places them in competition with themselves, and because it has proven to be attainable.

Accuracy is the third consideration; it is not a viable goal until technique and speed have been developed. While in the process of learning to keyboard, it is inevitable that some strokes will miss their mark. A "miss" is not a mistake; it is a 'misstroke.' That is the business education term; it does not imply frustration. Students should not disrupt hand-finger position to correct misstrokes during lessons. Once technique and speed are established, effort devoted to developing accuracy will yield quick results.

WHAT INSTRUCTIONAL STRATEGIES ARE APPROPRIATE?

Keyboarding has its specific instructional requirements: modeling, reinforcement, pacing, differentiated practice, motivation, and an easy to use program (Sormunen, 1984). A combination of Computer-Assisted-Instruction (CAI) software and a keyboarding textbook with coordinated lessons provide the most desirable instructional media (Jackson and Berg, 1986).

The elementary keyboarding curriculum should include materials written at an appropriate reading level (about 4th grade). Technique should be graphically illustrated, with new key learning activities demonstrated with an illustration showing location and proper reach technique. Real words should be introduced as soon as possible (Brady, 1985, Sormunen, 1984). New key presentation should be based on solid criteria including ease of operation and frequency of use. Approximately two keys should be taught in one lesson. There should be frequent review without new learning. The goals of practice should be clearly identified. There should be frequent monitoring of technique (Robinson, 1985).

WHO SHOULD TEACH KEYBOARDING IN THE ELEMENTARY SCHOOL?

Keyboarding in the elementary school should be taught by elementary teachers (Wetzel, 1985, Cameron, 1986). Students should be taught by someone who is trained to teach keyboarding. Regular classroom teachers, who are not touch typists themselves, can teach keyboarding with minimal training. The training is essential, however, and should include the principles of skill building and the correct method of teaching keyboarding. A viable approach to provide this training is to set up a curriculum council that will allow certified typing teachers to coordinate their efforts and expertise with elementary teachers (Erthal,1985).

WHAT IS THE TEACHER'S ROLE?

The teacher is the most important component of any keyboarding program. The teacher directs the lessons; modeling, dictating, providing reinforcement, and pacing. Evaluation, focusing on correct technique, is critical and can only be accomplished by the teacher observing the students as they key. Immediate feedback for the students on technique is essential.

WHERE DOES KEYBOARDING FIT INTO THE CURRICULUM?

If students are going to use the computer for word processing or programming, they will require a substantial degree of keyboarding skills (Wetzel, 1985). Keyboarding does not have to replace something that is already taught, but could be incorporated into existing instruction. One area of the curriculum that could include keyboarding is language arts, especially during handwriting (McGarvey, 1986). For example, some of the time allotted handwriting could be devoted to keyboarding.

Keyboarding instruction requires a daily block of 20 to 30 minutes, over a period of four to six weeks. It is an investment that will pay off as students gain facility in keyboarding and are able to use the computer more productively. This is an important consideration given the bulging elementary curriculum (Minnesota Curriculum Services Center, 1984).

HOW DO YOU ORGANIZE YOUR CLASS FOR KEYBOARDING?

The ideal situation for teaching keyboarding would be with one computer for each child in a computer lab, which would allow time for a complete lesson to be taught and time for practice. This is the most effective use of computer time (Wetzel, 1985).

Options or variations of the ideal situation abound:

- If the lab cannot be used every day the lesson could be taught in the lab and the practice could be done in the classroom.
- Another option would be a computer for every two children, alternating between the keyboard and the paper template. The student who is not on the computer would be using a paper keyboard template to do the lesson.
- If the option exists to share the class with another teacher, half of the class could be at the computers and the other half working away fron the computers.
- Practice could be done in the classroom on one computer, with each student scheduled at the computer for a 5-10 minute session.
- A less effective way of teaching the lesson would be for the whole class to use paper templates and then cycle through the computer for practice. However, the paper templates do not provide the necessary kinesthetic feedback.

The important thing to remember when trying to schedule computer time, is that all students get a turn at the computer every day (Wetzel, 1985).

WHAT SHOULD YOU LOOK FOR IN GOOD KEYBOARDING SOFTWARE?

According to Jackson and Berg (1986), keyboarding software is becoming more sophisticated, including instruction in correct technique, paced skill development, speed reports, continuous positive feedback, motivational devices such as games and printable progress reports, and an option for open screen typing which can later be printed. Other criteria for good software follows the components of a good keyboarding program. The software should emphasize technique, speed, and accuracy in that order. It should review correct body position. On screen prompts should be included to aid students in technique. Real words should be introduced as soon as possible. It should alternate the use of the hands. New material should be presented in small increments. Skills should be reviewed regularly. Rate information should be provided. The material should be at an appropriate grade level (Robinson, 1985).

Keyboard software is not mandatory in the initial instruction; rather, a word processing program can be used effectively for teacher-directed lessons. Keyboard software has an appropriate place following the establishment of keyboarding technique because even the best keyboard software emphasizes speed to some degree. Many teachers have appropriately postponed use of keyboard software until students have established the technique and are ready for speed.

All of these issues reflect the overriding theme that word processing and keyboarding are bound together in the curriculum. Within keyboarding, technique is the obvious first focus. It has become clear to us over the past few years that keyboarding is a fundamental component in writing with computers. Without proper keyboarding instruction students will fail when they attempt to write with computers. There is definitely a place for keyboarding instruction within the language arts curriculum. Effective programs for teaching keyboarding are available and students are capable of learning this timely and valuable skill.

References:

Bartholome, L. "Keyboarding - A Need in Today's Classroom". Printout, 2,1 (Fall, 1984), pp. 1 & 3.

Brady, J. "Who Needs Ten-Finger Typing?". Classroom Computer Learning. (September, 1984), pp. 64-69.

Cameron, John "Keyboarding in the Elementary School". Computers In Education, (March, 1986).

Cowels, Millie, and Others. "An Analysis of Young Children Learning Keyboarding Skills". Paper, University of Alabama in Birmingham, 1983. 15 pages.

Erthal, Margaret J. "The Status of Keyboarding". Journal of Business Education. (February, 1985), pp. 192-193.

Fidanque, Ann and Sullivan, Gail "Word Processing: Writing With Computers". Eugene School District 4J Proposed Computer Curriculum. (December, 1985).

Hansen, Karen "Guidelines For The Instruction Of Keyboarding". Puyallup School District, Washington, (April, 1985), 12 pages.

Hedley, Martha Kay "Once More: The Keyboarding Debate". Business Exchange, (October, 1985), pp. 18-19.

Jackson & Berg "Elementary Keyboarding - Is It Important?". The Computing Teacher, (March, 1986), pp. 10-11.

Kisner, E. "Keyboarding - A Must in Tomorrow's World". The Computing Teacher, (February, 1984), pp. 21-22.

McGarvey, Jack, "Is It Time to Boot Out Cursive Writing?". Classroom Computer Learning, 6, 6 (March 6, 1986), pp. 36-37.

Minnesota Curriculum Services Center. "The Electronic Keyboard". (1984), pp. 1-7.

Robinson, J. W. "Elementary Keyboarding: Some Factors to Consider". Century 21 Reporter, (Spring, 1985), pp. 2-4.

Rhodes, George S. "Keyboarding - The Keystone of Computer Literacy". The Balance Sheet, 67, 4 (March/April, 1986).

Rowe, John L. "An Experiment in Teaching Typewriting to Third and Fourth Grade Students". College of Education of the University of North Dakota Record, XLIV (November, 1958).

Sormunen, Carolee "Inservice Workshops: One Answer to the Issue Of Elementary School Keyboarding". Journal of Business Education, (October, 1984).

Wetzel, K. "Keyboarding Skills: Elementary My Dear Teacher?". The Computing Teacher, 12,9 (June, 1985) pp. 15-19.

Wood, Ben and Freeman, Frank, An Experiment Study of the Educational Influences of the Typewriter in the Elementary School. New York, N.Y.Macmillan, 1932.

MAKING THE COMPUTER SPEAK THE CHILD'S LANGUAGE

Larry Flick

The clinical research technique has shown that sensitivity to children's language reveals a depth of meaning to children's thoughts over a wide range of natural phenomena (Piaget, 1929). Current research in science and social science education shows the powerful influence on instruction of the meaning children bring to the classroom (Novak and Gowin, 1984). It is clear that classroom instruction must include techniques for exploring and clarifying implicit meaning of instructional terms held by students (Osborne & Freyberg, 1985).

Elementary education majors at the undergraduate level were given the problem of exploring mathematical concepts through computer representations. Results suggested that interaction with computer representations can challenge currently held concepts and broaden the understanding of others.

Each student identified a math concept he/she wished to explore. This led to a consideration of the educational problems associated with teaching this concept. Next, each settled on a way of representing the idea using Logo turtle graphics (Davidson, 1982). From there the representation was picked apart in order to identify ways of manipulating important aspects of the representation.

These college students were asked to consider their computer project as a focus of discussion in the elementary classroom. At the same time, they were to look for ways that student interaction with it could be used to further understanding of the chosen concept.

The programming language Logo was used since it allows the naming of procedures with words. Further, the word associated with a given procedure can be changed to accommodate a variety of interpretations by different students. In this way a problem or topic can be presented using meaningfully named procedures. Students can use these procedures and rename them as their understanding of the problem develops. The following examples will make these ideas explicit.

Turtle Graphics in Logo

FORWARD 30 is the Logo command to move the turtle 30 steps in the direction in which it is currently pointing. When the turtle is pointing toward the top on the monitor screen, with respect to the child's body, it is going up and not forward. One solution is to define a procedure, UP, which always moves the turtle at a heading of 0, which is toward the top of the screen. DOWN, RIGHT, and LEFT can be developed in the same way for headings of 180, 90, and 270 respectively. This restricted form of turtle motion will become limiting when more complex problems are encountered.

The problem of guidance with respect to compass direction requires more sophisticated turtle motion. Students studying the Age of Exploration could be set the task of developing methods for navigation at sea. Armed with a handout about turtle guidance (SETHEADING, HEADING, TOWARDS, POSITION, and SETPOSITION), they could be charged with establishing methods for people guidance in a non-computer environment.

The computer techniques embedded in turtle graphics are well named for the above problem, but the educational value does not reside in the Logo commands. The value is in the development of meaning about a problem stimulated by instruction. How, for instance, does a ship determine its "bearing" without the turtle function: HEADING? Students common sense understanding of navigation can be stimulated by setting familiar landmarks around the computer screen. Using SETHEADING TOWARDS the landmark they can define navigational procedures in theirown terms, i.e. WHICH.WAY.TO may be the title of a procedure which takes a landmark position input and returns the compass heading.

Dealing with ideosyncratic terms for instructional concepts requires skilled perception and sensitivity to linguistic meaning on the part of the teacher. By making the computer speak their own language, the prospective elementary school teachers in this study gained insight into the limits of their own conceptions when explaining some mathematical ideas.

Roundness

A circle does not conform to the standard methods of constructing and measuring polygons. Students have to use different equipment to draw them and different words to describe them. The area and perimeter of polygons can be explained by measures which are directly observable. There may be some intermediate steps that require dividing the shape into squares or triangles, but the process is understandable from intuitively simple, linear measurements. Therefore, defining the dimensions of a circle is difficult because of what children have learned about other figures.

The elementary education majors who investigated the circle had not reconciled their understanding of circle with other figures. They worked from the basic circle procedure often discovered by children using Logo: REPEAT 360 [FD 1 RT 1]. The most striking feature to the university students was the relationship between numbers 360 and the "right turn" of 1. Any pair of factors of 360 will produce a closed figure. This became known as a Total Turtle Trip or TTT (Abelson & diSessa, 1980). Exploiting this feature of the turtle circle one student produced:

```
TO SHAPE.A :N
REPEAT :N [FD 5  RT (360/:N)]
END

TO SHAPE.B :ANGLE
REPEAT (360/:ANGLE) [FD 5  RT :ANGLE]
END
```

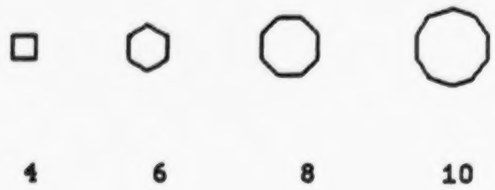
These shapes are curious in that each has one input and each can draw circles, but in different ways. SHAPE.A produces a larger "circle" with increasing :N while SHAPE.B produces a smaller "circle" with increasing :ANGLE (see Figure 1). Explaining what the input numbers are "measuring" in these circles became a topic for investigation.

SHAPE.A has another feature that becomes noticeable with small :N. SHAPE.A 4, for instance, draws a square with sides of 5 units on the monitor. SHAPE.A 6 draws a hexagon, an input of 8 results in an octagon. For inputs larger than about 10, SHAPE.A begins to be noticed as a circle. The input to SHAPE.A is measuring something about circles that is related to other figures. The inverse is true for SHAPE.B (see Figure 1).

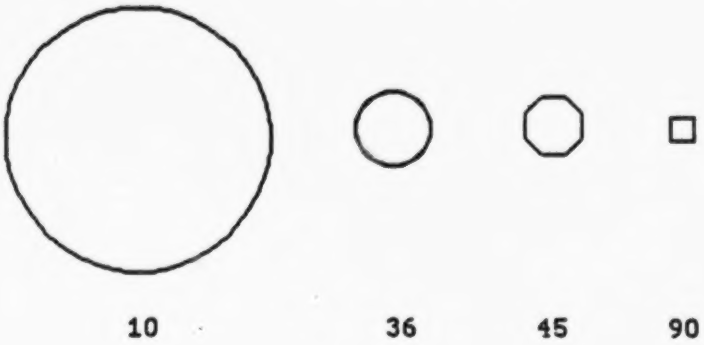
When used with one fifth grade boy, he remarked excitedly "I made a square but now it's a circle!" The question which has been posed through SHAPE.A 4 is: What input(s) to SHAPE.A make a circle and why? This anomaly of a procedure which draws both a square and circle was confronted uncomfortably by the college students while working on their projects. "It moves so often and slightly that it appears to be a 'real' circle. But, it can only approximate a 'true' circle with straight lines."

The language of the fifth grade boy expresses a notion about circles that does not come out of standard classroom discussions and activities with circles. The university student reveals that he believes that "real" and "true" circles are unrelated to straight lines. Another university student expressed it this way "The smaller the degree of turn, the more rounded the shape. A square with only 4 turns (of 90° each) is not as round as a dodecahedron." At what point would a child label a shape as "round"? More importantly, how would that child reconcile their concept of round with their concept of straight?

Figure 1



SHAPE.A :N



SHAPE.B :ANGLE

Inputs to SHAPE.A show progression through regular polygons towards circle-type shape. SHAPE.B show inverse progression towards the shape of a square.

Understanding curvature is crucial to understanding various science topics, such as, Newtonian dynamics (diSessa, 1982; Flick, 1985). The instructional contexts being designed by these students confronts the concept of "roundness" in a way not seen in standard classroom settings. The next set of procedures by another university student explored this same question:

```
TO SHAPE.C  :N  :ANGLE
REPEAT  :N  [FD 10  RT :ANGLE]
END

TO SHAPE.D  :D
REPEAT  180  [FD :D  RT 3]
END
```

SHAPE.C opens for explicit investigation that which is implicit in SHAPE.A and SHAPE.B. SHAPE.C highlights the "turning" aspect of a circle. Why, for instance, do certain inputs to SHAPE.C produce only part of a circle? For some inputs SHAPE.C retraces itself thus drawing more than one circle. This was an unexpected and unwanted result for the student who wrote SHAPE.C. Her concept of 'circle' blocked herown understanding of her Logo procedures. SHAPE.C was psychologically imbued with her 'circle' concept, but her Logo code was not so constrained and more open to interpretation. She felt that this more open interpretation interfered with the educational aspects of her Logo procedures. Interaction with the computer and her classmates forced further consideration of her own idea of 'circle' and its educational implications.

Unexpected results are the stimulus for new questions, the precursors to new knowledge. Being able to draw part of a circle entails the idea of drawing over the same circle more than once. This resulted in one student commenting "I think there is an inverse relation between the REPEAT number (:N) and the bracketed commands (FD 10 RT :ANGLE). How do you calculate half-a-circle base-length (diameter) mathematically?" The change in perspective resulting from this student's manipulation of a SHAPE.C-type procedure produced a deeper understanding of the 'diameter' concept (half-a-circle base-length). Put another way her question is "Without knowing any more about a circle than I do a square, how do you determine the distance accross?" This information is distributed in school in the form of "pi", stripped of the geometrical problem for which it was a solution.

Discussions with the Computer

The SHAPE procedures are an example of how representations, when used as models for a specific concept, stimulate the need for explanations. Elementary school children are skilled at creating explanations. Dealing with the explanations of individual students may be impractical in a whole class setting. Using a computer in a way that makes it adaptable to children's language can create an effective environment for unpacking the meaning behind children's explanations. The SHAPE procedures demonstrated three ways this worked with the elementary education students:

1. **The critical parameters of the problem can be labeled in a way that reflects the interpretation of the student.** Students are free to label the input numbers with whatever word or phrase suits their interpretation. The "shapes" themselves can be named with reference to the images they inspire. SHAPE.C, for instance, could be thought of as a SPOOL.AND.THREAD. The input :N increases the length of the thread and :ANGLE determines the size of the spool. When SPOOL.AND.THREAD runs, the turtle wraps the thread around the spool. The new and potentially confusing term 'circumference' is not initally needed. The problem becomes, Determine the length (:N) of thread and size (:ANGLE) of spool such that the thread wraps around the spool once. There are, of course, many answers, but they all must satisfy TTT: (:N X :ANGLE) = 360.

2. **Use of common terminology during instruction promotes transfer to common examples.** The formalized expression: $C = \Pi \times D$ depends upon a truely strange number and a line which is not part of the figure drawn through an invisable point. The spool and thread problem could be posed with objects alone but the answer is trivial and is not easily associated with numbers. The visual answer derived from cutting thread does not convey the constant relation between "distance around" and "distance through". The problem could be posed with objects but the mathematical significance is demonstrated with the computer.

3. **New terminology can be introduced in relation to intuitive concepts which have a numerical base.** "Distance around" is already used as the root concept for "circumference" and "perimeter". However, there remains the task of establishing a numerical component for this concept. The input :N in SHAPE.A produces larger circles by turning less with each step, therefore, taking more steps to complete the circle. Input :D in SHAPE.D increases the size of the circle by taking larger steps with each turn. Circumference is associated with both the number and size of the steps

around the circle. This is consistent with "distance around" in other shapes. "Pi" and "diameter" will remain enigmas as long as there is no counterpart in other "regular" figures. The relationship between diagonals and perimeter in other figures has no major place in standard math instruction. The computer can focus interest in this relationship within other regular figures.

Conclusion

There is an important link between the words students use to describe an idea and how they will use that idea to solve problems. When instruction inserts a new term into a student's vocabulary via a definition, there is a risk that the deeper meaning rooted in practical knowledge will get distorted. This makes the knowledge ineffective in solving spontaneous problems and relagates the new term to "school work."

Computer interaction can be made adaptive to student interpretation through a computing environment such as Logo. Through the use of words as names for computer procedures and the skillful structuring of the experience by the teacher, students can make theirown ideas explicit. Explicit knowledge is shareable and shareable knowledge is more easily enriched by the surrounding ideas.

The computer is also flexable. Once defined, a procedure can be "stretched" in many ways through the use of variable inputs. Different size figures in different orientations are like a looseleaf workbook with lots of plastic overlays. Students can diverge and create and yet, embedded in this process is the pathway to the focus problem. The computer is a piece of lab equipment which exercises both the students' visual and planning skills as well as their skill in creating and explaining terms.

References

Abelson, Harold & diSessa, Andrea, Turtle Geometry. Cambridge, Massachusetts: The MIT Press 1980. 477 pages.

Davidson, Laurence J. Apple Logo Reference Manual. Pt. Claire, Quebec, Canada: Logo Computer Systems 1982. 186 pages.

diSessa, Andrea "Unlearning Aristotelian Physics." Cognitive Science, 6 (1982), pp. 37-75.

Flick, Lawrence, Natural Reasoning in children: An Analysis of Interview and Keyboard Behavior in Computer-Simulated Force and Motion Problems. Ann Arbor, Michigan: University Microfilms International 1985. 333 pages.

Novak, Joseph D. & Gowin, D. Bob, Learning how to Learn. Cambridge, Massachusetts: Cambridge University Press 1984. 199 pages.

Osborne, Roger & Freyberg, Peter, Learning in Science: The Implications of Children's Science. Portsmouth, New Hampshire: Heinemann Publishers 1985. 198 pages.

Piaget, Jean, The Child's Conception of the World. London: Routledge & Kegan Paul, Ltd. 1929. 490 pages.

GOING ONLINE: DIALING INTO THE FUTURE

Deborah L. Graham, M.L.S.

A Word Before We Begin:

The most useful class I ever had in college was a course on teaching adult remedial reading. The class was valuable not so much for the actual content, but because the professor was actually teaching a class on "How to Teach". She emphasized one point over and over again - that anything we teach should have useful purpose and meaning for the student. The same point applies to the use of online systems. Be sure that you have very legitimate needs that such systems will meet before you commit time and funding to online services.

To phrase it another way, we have two basic policies at our Library at Sacred Heart. 1) Never MAKE policies that violate human nature! and 2) Never let policy violate common sense. Or, 1) if you don't want students to push a system to its limits, you had better not even introduce it to them. It is the nature of students to push things to their limits. And, 2) if the manual says something can't be done, and students are obviously doing it, forget the manual.

These statements may all seem terribly obvious. But we are about to go into the labyrinth of online systems, and it is very easy to lose sight of common sense and 'the obvious' once enmeshed in the online maze.

About Costs:

One of the first things new online users notice is that online systems cost money. This seems to come as a shock to many individuals in the information and library professions. But it shouldn't. Information services have ALWAYS cost money - usually in personnel time. And the major contribution that present online services offer is savings in personnel time. The pitfall in online services is the danger of misdirected activity. Approaching a system from the wrong mind-set, or choosing the wrong system as a source of data can be both frustrating and expensive. The same errors have happened for years in our traditional print systems. The difference is that in online systems, the true dollar cost is more obvious, and more immediately apparent.

To protect against unnecessary costs in the use of online services, follow these guidelines:

1) For the actual computer interaction, know what you want before you go online. This is not always possible in complex scientific searches, but for the most part some time spent really analyzing the question will result in a less expensive answer.

2) In purchasing or selecting online services to which you want to subscribe, have in mind exactly what you want the system or service to do: don't allow the industry advertising to tell you want you want. Always approach the purchase and use of online services from the point of view of WHAT YOU WANT not from WHAT YOU THINK YOU CAN GET. This is critical, as your purpose will affect the selection of hardware and software, as well as the dial access systems to which you choose to subscribe.

3) When you do sign up online, be prepared to spend some money on "research & development"; on simply learning your way around the system. And order the manuals, even if there is an additional charge. If you don't invest the time and effort to really learn the system you choose, it will never really produce the results that it should for you. Budget for online exploration time.

What Can I Do Online?

Since the first objective in going "online" is to decide what you want to do, we do need an overview of some of the possibilities online or "dial-up access" computing can offer. We are using the term 'online' to describe dial-up systems. This is not technically the true meaning of 'online', but the word has come to mean that for popular or lay-level computing. Fred Hoyle, the astronomer and science fiction author, said that "Space is STRANGER than we can imagine". Online services offer the same sort of challenge. Most online systems today mimic existing paper or print oriented systems. But many offer opportunities to actually invent new forms of information exchange and knowledge transfer. Try not to think in terms of print systems when you approach online systems. Keep in mind things you would like to do that print systems have not permitted, that might now be possible. Students and young people will have an advantage over some of us here. Since they do not know what they can't do, or what traditional information organization and access patterns have been, they explore new options based on what they want. This should make online adventures very exciting for those of you who teach.

The "traditional" patterned services include:

Dial-up Services: Bibliographic and hard databases. These are indexes and databases of facts that can be accessed over the phone from large computer systems. They are "electronic" equivalents of traditional indexes and reference books. They do have many advantages, as the indexing of an article or book can be extended to actual words in the text, not just to broad "controlled" subject headings. In that respect, they are more efficient than the print systems. The principles of use are the same, however. And most of the time, the requestor must still get a copy of some printed document to complete his or her search for answers. Dial-up databases are provided mainly through large brokered systems, such as DIALOG, and BRS. The subscriber calls these systems through one of the major national networks, chooses a specific database, and locates references by giving specific commands to the selected program. Commands vary from system to system, and sometimes from database to database within a system. Costs may also vary widely, depending on the database selected.

Some major system brokers are now offering "user friendly" front-end programs. These are usually a series of menus that allow the requestor to select options, rather than learning the specific system or database commands. Unlike "open" systems, where the requestor can exercise any of the options programmed into a system, the "user-friendly" or "front-end" programming generally limits the ability to refine and pinpoint specific textwords or concepts. For some systems, where broad overviews or general subjects are desired, the "front-end" programs are more than adequate and provide an easy way to get introduced to online access. They tend to be slightly more expensive than access by a skilled trained searcher who is familiar with a system's commands; but they are definitely cheaper for the novice or infrequent searcher who is not doing intricate research.

The print-oriented systems generally provide the following types of access to data:

a. textwords - actual words found in the title or abstract of a magazine article or the words in the text of a book.

b. keywords and/or controlled vocabularies - "subject headings" which group items on a similar topic together, regardless of the words in a title or text.

c. author access.

d. source or publication title access

Once located by the above techniques, requestors can narrow the scope of a search by date and other parameters. The problem is usually too much data, not too little, and the ability to narrow a search becomes very important.

WARNING: Systems also have what we can"unwritten sets", or biases, that affect the way materials are indexed and/or the way commands actually execute. Knowing (or not knowing) these idiosyncrasies of a system will result in very different online search retrievals.

Not all traditional type systems are bibliographic. Some feature actual data that can be called up or that you enter into the system, that is later used to perform print oriented functions. On some systems (such as OCLC), the user must enter data in response to ques generated by the online system. These "real-time" data entry systems are slower than systems that allow free text entry and/or uploading (transmitting files directly from your computer to the other one). The advantage to such systems lies in sophisticated programming that allows special manipulation of the data you are entering as you use them. These special applications can include the production of catalog cards, statistical records, magazine issue claim forms, or any number of things. When signing up with any system you'll want to determine if it is a controlled data-entry system, or free-form system, and if the data entry style suits your needs.

Non-traditional services include:

1) Electronic mail - "closed" and open systems: these can be in imitation of print or provide opportunities for "third wave" uses.

2) Bulletin board systems - related to electronic mail, but slightly different, these systems can provide new configurations of "invisible colleges" (or kindergartens). These can be controlled data entry systems or free-form.

3) Terminal to terminal transmissions. This area of online use is not yet very common. It combines traditional print and conversational telephone use patterns for a variety of applications. "Computer conferencing" is one application.

How Do I Get Online?

Hardware:

You can use either a microcomputer or a 'dumb' terminal to access dial-up computer systems. Microcomputers have some advantages, as they allow you to do some activities "offline" and 'upload' to systems. Since you pay a connect charge for every minute you spend online, if you can do things before you dial up, it saves money. You will also need a modem, an instrument that allows your machine to "talk" to other computers over the phone lines; and software to run your modem if you are using a microcomputer. (No software is needed with a 'dumb' terminal.) Your equipment should be ASCII compatible. If you live in a rural area, buy a GOOD modem. Cheap modems do not have adequate signal integrity to function well over long-distance lines.

CAUTION: The computer industry has in many instances "over-engineered" hardware. Select equipment that is appropriate to your needs. "Smart" equipment can be expensive and wasteful if you only want to be able to dial up any phone number. Error checking, and other sophisticated processes, are very valuable to programmers, but essentially useless for non-programmer online users dealing only with ASCII text files.

Each piece of equipment you purchase may have some form of "intelligence". When this is the case, you will have to read all the manuals and be certain that everything is set to the same parameters. This is much harder than it sounds, and accounts for most online start-up problems.

Software:

Telecommunication programs come in all sizes, shapes and operating systems. You will need to buy software that is compatible with your operating system. We have found the ASCOM program, available for several operating systems, to be the most flexible and easy to use of the present popular programs.

Telecommunications Skills/Trouble-shooting:

Now that you know what system you want, and have mastered your hardware and software, it is time to dial in. You will most likely call the computer service through one of the national telecommunications networks. These are special phone lines that provide a clearer signal from distant computers to your modem. A fee is charged by your system vendor, part of which goes to these special phone networks for the use of their lines. Each national network varies slightly in "login" procedures. All provide ways for you to let the host computer system know what type of equipment you are using to make a good "match" between your equipment and the database vendor. This involves adding control characters or "telecommunication pads" to the signals that come over the phone line. As long as you are using the correct ID codes, there should be no real problem using the national networks. Occasionally a burst of "phone line noise" (the Cosmic Voice) will come over the line.

Special line protectors can be purchased to prevent these noises if you prefer. If you don't have one of the protectors, you may receive random ASCII characters, often control (command) characters that will sometimes reset the parameters of your equipment. The Cosmic Voice can always be dealt with definitively by disconnecting from the phone, shutting down your equipment, and starting over. Usually you can just continue after the noise and eliminate the garbage characters later. When you do get line noise, you may experience difficulties in using word processing software with the file that was open at the time. Certain control characters produce in "black holes" in text files. If you encounter "black hole" problems, it is best to check around for someone with similar software who has already dealt with the situation. In fact, as a matter of policy, it will be a good idea to find some other individuals with similar equipment and software who can act as a support group. Local professional organizations and computer stores can often direct you to such user groups.

EXPANDING THE MIND

Now you can join the ranks of online explorers. While it takes time and effort to master equipment and software, I am sure you will find it worthwhile. The most important developments in online communication are yet to come. Printed publication and print based information management systems have provided us with a database of human thought and endeavor. And this database bias has governed the very way we think, learn, and approach problem-solving and decision-making. Online telecommunications, through variations on electronic mail, provide the ability to broadcast and refine ideas in cooperation with peers VERY rapidly. Our "human database" becomes instead a "human equation", a unified entity instead of a collection of fragments. The change in ideas and realities that will result from the speed of these human interactions is probably "stranger than we can imagine".

USEFUL PUBLIC DOMAIN SOFTWARE FOR EDUCATORS

Kurt Gustafson

The Educator's Problem

Educators of today are encouraged to make use of new tools as they become available. The most notable in recent years has been the micro-computer. A notable item about computers is that in order to use them educators must expend generous amounts of time becoming proficient with them. Also large portions of already sparse budgets must be used to obtain the software that will make the computer a useful tool in the business of teaching and learning. It is often discovered, too late, that costly commercial software does not meet the specific needs for which it was ordered. It then gathers dust on the shelf and money that could have been better spent is lost. Becoming proficient simply takes time and practice, but a partial solution to the financial difficulty of obtaining software is the software found in the many public domain sources.

What Is Public Domain Software?

In a word, public domain software is free. Sometimes a small charge is made for duplication and shipping, but it is usually less than $5.00 per disk and may be as little as $2.00. Like commercial software, public domain software can be obtained in every category including applications, games, nutrition, electronics, business, education, and utilities. They can perform almost any computer task so you and your computer can experiment in new applications that cost may have prevented you from before. Also like their commercial cousins, public domain software is of varying quality. For the most part public domain software lacks the thorough documentation that accompanies commercial programs. This is usually not a major obstacle because programs in the public domain tend not to be as complex, and perform only single tasks per program. Public domain disks nearly always contain several programs per disk (up to 30 or more!) of varying utility.

How To Get Public Domain Software.

There are many sources of PD software. On-line services such as the Source and CompuServe allow users to download programs by using modems. Magazines are also a good source of programs. You can type in the listings, or usually for a small fee a disk can be purchased from the publisher for the same program. Some good magazine sources are Nibble, Computist, A+, The Computing Teacher, and most other educational computing magazines.

Users' groups probably provide the most abundant source of PD software. Some users' groups charge non-members fees for software, and offer it free or at very low cost to members. Most large cities have users' groups organized around different makes of computers. These groups can be contacted through computer stores or through university or college computer departments.

Potentially larger PD sources are national users' groups such as the Big Red Apple Club (BRAC) of Norfolk, Nebraska. This group has an extensive library of PD software which is constantly expanding. Membership to BRAC is $12.00 per year and includes a monthly newsletter, Scarlett, which describes new PD software, offers technical help, tips, reviews, and educational applications of PD software. BRAC has a PD library of over 3000 programs which are available for $2.50 per disk (less if you order more than 10).

The International Apple Core (IAC), headquartered in San Jose, California also has a huge library of PD disks and will send you a descriptive catalog called Apple][Public-Domain Software Library Catalog, upon request. IAC charges $4.50 per disk and offers other services as well.

A hybrid public domain source is the Public Domain Exchange (PDE), from San Jose, California. This is a commercial public domain source that offers some special features others do not. PDE has a staff of reviewers that screen all incoming software before it is added to their library. Poor quality is weeded out and programs that are accepted are grouped together on categorical disks when possible. PDE publishes a newsletter which contains reviews, tips, and applications. In addition, a book, The Best of Apple Public Domain Software, has been published which categorizes and describes most of the best PD programs available.

What To Do When You Get Public Domain Software.

The first thing one must do upon receiving PD software is determine which ones will help you achieve instructional goals (content area programs), help you implement some task (application and utility programs) or fill some other need (entertainment?). To do this the user should pull up a comfortable chair in front of the computer, have paper and pencil handy, and systematically experiment with each of the programs in turn, to see features, strengths and weaknesses. Since printed documentation rarely accompanies PD software it is good practice to write down some operational procedures and general descriptive notes for each of the programs you survey. Usually the more complex PD programs will have documentation as part of the program, or as a separate text file that can be printed either from the program itself, or from a word processor.

Organization of Your Collection.

After you have begun to amass several sets of PD programs and program notes you will find it useful to organize your disks with some kind of numbering system, and assemble your notes in a three ring binder by categories and indexed with disk numbers. It is advisable to do this early in your collecting, because PD collections tend to grow quickly and it is easy to forget what a particular program did, or what disk it was stored on.

Some Notable Programs

FrEdWriter: A very good word-processor.

FrEdWriter is a program written by Al Rogers of the CUE Softswap Project. This program is an improved ProDOS version of Free-Writer, written by Paul Lutus of Apple Writer fame. This program is surprisingly powerful and yet it is easy to learn. Most of the feature commands use single letter codes that match the first letter of the function to be performed (ie. <N> is for New screen). FrEDWriter contains four text files of ample documentation which can be printed by FrEDWriter. At any time during operation of FrEDWriter the user may consult a built in tutorial that summarizes all commands. This program will operate in 40 or 80 column video display on Apple][,][+,][e, or][c. Included on the disk is a feature called 'prompt boxing' used to help young students in writing skills, or anything else you would like to prompt. An extensive set of prompt-box lessons is available from CUE Inc. Softswap. Spanish FrEDWriter information can be obtained from Hands-On Training Company as well. It is hard to believe that this program is free once its power is revealed!

DB Senior: A simple easy to learn and use data base program.

DB Senior is a powerful menu driven data base program with many features including alphabetical and numerical sorting, linear or binary searching, record deletion, and reversible scrolling. Included on the disk is a comprehensive manual which can be printed using a word processor that handles standard text files. DB Senior can be obtained from the Big Red Apple Club.

PS Graphic Grabber V3.0: The first and the best PSG converter.

The Print Shop Graphic Grabber by Ray Darrah is a companion utility for Broderbund's popular Print Shop. With the Graphic Grabber you can take portions of any previously captured high resolution graphic screen. This means that art work from the many drawing tablets, and graphic drawing programs can be used to add to your 'graphic' collection. The Graphic Grabber has the ability to stretch the X and/or Y axes of the capture window.

This allows the user to use up to an entire high resolution screen as a 'graphic'. Once the desired area is surrounded it is scrunched down to 'graphic size, and can be saved and used as is with Print Shop. Since some resolution is lost in the scrunching process, another feature allows the density of lit pixels to be intensified or diminished as you wish. The Graphic Grabber can be obtained from the Computer Learning Center.

SLALOM: Print your spread sheets vertically for less!

SLALOM allows the user to print wide spread sheets vertically rather than horizontally and this allows you to put away your scissors and tape for beautiful one piece sheets. SLALOM works with most of the popular printers, parallel or serial interface cards and with Apple][+,][e or][c. This program comes with written documentation. Both the program and the documentation are available from the Big Red Apple Club.

EAMON Adventures: 'Want to leave town, or learn map skills?

The EAMON Adventure series has over 100 disks, each containing an adventure scenario. In addition to great fantasy role playing, one must use careful record keeping and develop accurate map drawing and reading skills. Playing these adventures is only the beginning of the fun and learning though. Eamon also has a scenario designer disk which allows you to create any world, ecosystem, or society your imagination can dream up. Your scenario can be realistic, fantastic, serious, or funny; the choice is yours. All of this is possible through a powerful menu driven program whose features your description and choice of up to a hundred places, things, special effects of things, and people/animals. Through your imagination and the easy to us[illegible] editor you can create scenarios to fit historical events, futur[illegible] events, environmental/ecological situations. Simply put, thi[illegible] scenario designer is an advanced simulation designer. EAMON is available from any of the public domain sources including Big Red Apple and the Computer Learning Center.

The programs described above are only a tiny fraction of the thousands available in the public domain. While only a few of these programs boast all the bells and whistles of their commercial counterparts, nearly every one will fit a niche and fulfill a need somewhere. Listed below you will find the names and addresses of a few of the more popular public domain sources.

Sources

The Big Red Apple Club
1105 S 13th, Suite 103
Norfolk, Nebraska 68701
(402) 379-4680

International Apple Core
Public Domain Library
908 George Street
Santa Clara, CA 95054
(408) 262-9114

Apple-Dayton
Box 1666
Fairborn, OH 45324
( )

Apple Avocation Alliance
1803 Warren Avenue
Cheyenne, WY 82001
(307) 632-8561

CUE Softswap Project
SMERC Library & Micro Ctr.
San Mateo Office of Edu.
333 Main Street
Redwood City, CA 94063

The Public Domain Exchange
673 Hermitage Place
San Jose, California 95134
(408) 942-0309

Computer Learning Center
P O Box 110876
Tacoma, Washington 98411
(206) 474-5808

Computer Budget Shopper
2203 Park Avenue O.V.
Cheyenne, WY 82007
(307) 635-2173

Pandora Software
P O Box 590
Clearfield, UT 84015
( )

Hands-On Training Company
4021 Allen School Rd.
Bonita, CA 92002

References

Freiberger, Paul "Cheap Thrills: A Great Selection of Public-Domain Software for Just Pennies" A+, 4, 8 (August 1986), pp. 30-34

Lathrop, Ann "San Mateo County Office of Education & Computer-Using Educators SOFTSWAP" The Computing Teacher, (April 1983), pp. 20-24.

Mace, Scott "Taking Software Out for a Spin" InfoWorld, 8, 27 (July 7 1986), pp. 27-28

ADVANCED TECHNIQUES USED IN FUNCTION GRAPHING

Dr. Lee Thomas Hill

One of the most useful and interesting applications of the microcomputer in an advanced mathematics course is that of graphing functions. To be sure, there are many other applications for microcomputers in mathematics, but none seems quite as satisfying as being able to quickly display a complex function graph. In fact, the ability to graph a function is often the first step in understanding more advanced concepts. The graph of the derivative relates directly to the max and min of a function. To understand the definite integral, it is helpful to sum the rectangles inscribed under a function's graph. A graph nicely displays the path of a particle. Fortunately for mathematicians, most microcomputers support high level graphing commands in BASIC. Thus, any instructor with a minimal programming knowledge should be able to use this important tool. The rest of this article is directed towards presenting techniques which enhance one's ability to graph functions using microcomputers. Since the Apple II family of computers is so pervasive in education, all algorithms presented will be described in Applesoft. Easy conversion to other computers and their BASICs should be possible.

The Basic Graphing Techniques

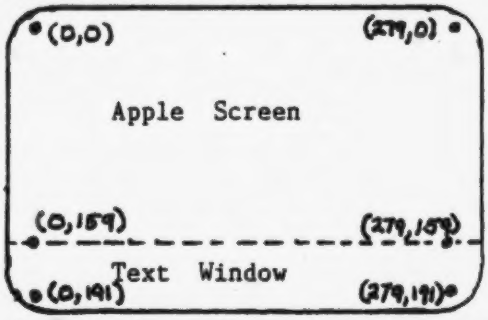
Since the subject of this article is 'Advanced Graphing Techniques', the 'basics' will only be mentioned. They are included for completeness and to create the environment for the advanced topics. For a more detailed development of the 'basics', the reader is referred to the this author's paper in the reading list.

There are three problems that must be solved before graphing can take place. These are: scaling, clipping, and an efficient algorithm for generating and graphing function values. In order to solve these problems, the domain and the range of the graph must be determined. The terms domain and range refer to the part of the function that will be displayed rather than their usual definitions from mathematics. Since interactive programs are easy to write and are more versatile, it is assumed that the information for the domain and range is entered during the execution of the program (i.e. interactively.)

Scaling

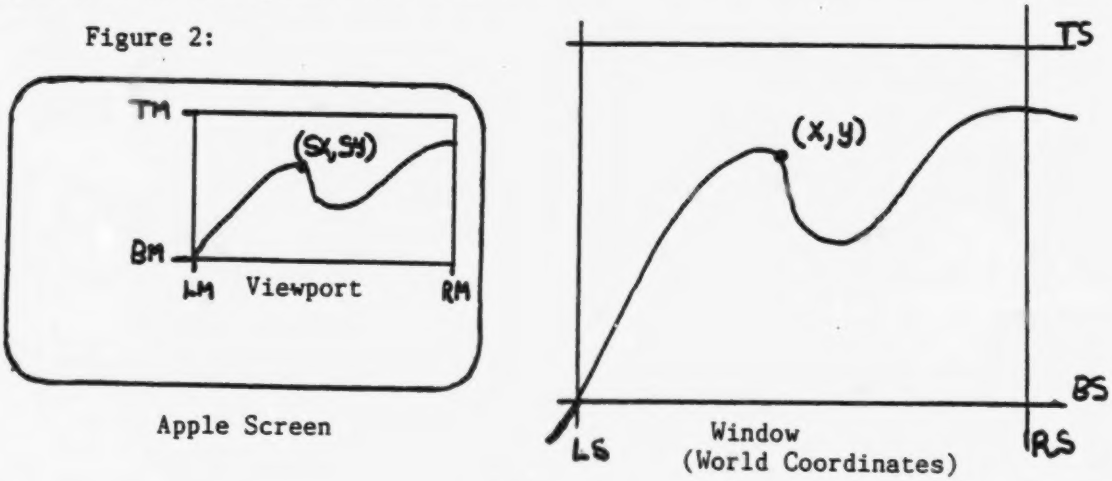
Scaling is the process of mapping a window determined by the domain and the range onto a region (viewport) of the graphics screen. Since we are using the Apple to illustrate our algorithms, it is important to know the dimensions of the Apple high resolution screen. There are approximately 6000 points on the Apple screen that can be addressed. Each of these points is called a 'pixel', short for picture element. The Apple screen is presented in figure 1. The coordinate pair for the upper left most pixel is (0,0). The first coordinate always refers to the displacement from the left edge of the screen and the second refers to the displacement from the top. The coordinate pair for the lower right most pixel is (279,191). It is possible to have four lines of text placed at the bottom of the Apple screen in which case the lower right most coordinate pair would be (279,159).

Figure 1:



We often label a pixel on the Apple screen with the coordinate pair (SX,SY). The 'S' stands for 'screen' coordinates and is used to distinguish them from their related pair (X,Y) which satisfies the function (i.e. Y = f(X).) The coordinate pair (X,Y) labels a point in the window and will be called world coordinates. In figure 2 a window and a viewport are shown with their vertical and horizontal edges labeled. The labels given in figure 2 will be referred to throughout this article.

Figure 2:



Apple Screen

Window
(World Coordinates)

ls: X coordinate of left edge of the domain.
rs: X coordinate of right edge of the domain.
ts: Y coordinate of top edge of the range.
bs: Y coordinate of bottom edge of the range.

lm: X screen coordinate of left edge of viewport.
rm: X screen coordinate of right edge of viewport.
tm: Y screen coordinate of top edge of viewport.
bm: Y screen coordinate of bottom edge of viewport.

A scaling formula states the functional relationship between the screen coordinates (SX,SY) and the world coordinates (X,Y). These formulas are given below.

```
SX = (RM-LM)*(X-LS)/(RS-LS) + LM

SY = BM - (BM-TM)*(Y-BS)/(TS-BS)
```

Since these formulas are often used it is wise to define them as functions in your BASIC program. For example:

```
DEF FNY(Y)=BM-(BM-TM)*(Y-BS)/(TS-BS)
```

Then in any application we could write SY=FNY(Y).

Clipping

Clipping is the process of determining when a point or line is outside of the window. In the case where a given point satisfying the function is outside of the window boundaries, the scaled screen image of that point should not be drawn. The clipping of an arbitrary line segment is quite involved and very time consuming (CPU-wise). Several algorithms are available for clipping line segments and can be found in the readings. Due to time considerations, it is always wise to avoid the problem of clipping whenever possible. You can avoid clipping by setting up your graphing problem in the right way. This will be accomplished in your graphing algorithm so nothing more need be said about clipping!

A Graphing Algorithm

The resolution of a function graph is determined by the number of points plotted across the screen. We will assume that the graph always moves from left to right within the viewport. The maximum resolution is the number of pixels across the viewport. This number is always one more than:

```
RES = RM - LM
```

We will choose RES to represent the number of gaps between the pixels rather than the number of pixels. The reason for this will be obvious later. It is possible to choose a smaller number for RES but any larger number would be useless. The smaller the RES, the faster the graphing! However, the smaller the RES, the less detailed the graph is. Most people prefer the added detail to the few seconds saved in graphing. We will assume that RES is a maximum and given by the formula above.

Related to RES is DX. DX represents the change in X from one function value to the next. Note that this corresponds to the change in X needed to span the gaps between pixels in the viewport! The formula for DX is:

```
DX = (RS-LS)/RES
```

All we need now is a function! We could place the function formula in the plotting algorithm but it is better to situate it near the beginning of the program using a function definition. This will enable you to find it and change it quickly if you wish to apply the graphing program to another function. An example of how this is done is illustrated below:

```
10 DEF FNF(X) = SIN(X)
```

If we wish to change the function, we change only line 10!

With these preliminaries taken care of we can now graph a function. Consider the following algorithm:

```
100 REM FUNCTION GRAPHER
110 HPLOT LM,TM TO RM,TM TO RM,BM TO LM,BM TO LM,TM
120 DX=(RS-LS)/RES: M=.7*(TS-BS): AV=(TS+BS)/2: OY=AV
130 ONERR GOTO200: E=1
140 X=LS: SX=LM
150 Y=FNF(X): SY=FNY(Y): A=(M<ABS(Y-OY))
160 IF E=1 THEN HPLOT SX,SY: E=0: GOTO 210
170 IF A=0 THEN HPLOT TO
    SX,SY*((Y>=BS)AND(Y<=TS))+TM*(Y>TS)+BM*(Y<BS):
    GOTO210
180 HPLOT TO SX-1,BM*(OY<AV)+TM*(OY>=AV)
190 HPLOT SX,BM*(Y<AV)+TM(Y>=AV): A=0: GOTO 170
200 E=1
210 OY=Y: SX=SX+1: X=X+DX: IF X<=RS THEN 150
220 POKE 216,0
```

The algorithm given above automatically takes care of the clipping problem. No pair (X,Y) to the left or the right of the window will be produced since the first value of X is LS and the X values increase until they reach RS. This means that LS ≤ X ≤ RS. If a point is above or below the window, line 170 takes care of it. This is done by the expression:

SY*((Y>=BS)AND(Y<=TS))+TM*(Y>TS)+BM*(Y<BS)

If Y is between the top and the bottom of the window, it is mapped into the viewport using the scaling function, if not, it is either assigned to the top edge or the bottom edge of the viewport depending on whether Y is above or below the window respectively. Line 110 draws a border around the viewport which covers up the function values plotted along the top or the bottom edge. The loop from line 150 to line 210 is executed RES+1 times. Each time the X value is incremented by DX and a new function value Y=FNF(X) is computed. The Y value is scaled by SY=FNY(Y) but the SX value is not. SX is merely incremented by one (line 210) since we are plotting at every pixel in the viewport. If we decide to change the resolution to something less then (RM-LM) then we must also adjust the increment for SX! The ONERR (line 130) is an Apple feature which allows us to detect function values which are undefined. This is a feature not available on all computers and could be left out with little effect. If it is used (on an Apple) then the algorithm should not be written as a subroutine (i.e. GOSUB 100). The reason for this is that the return address stack is scrambled when ONERR is used.

This function plotter is slightly more than a basic algorithm. Included is the ability to detect and properly graph vertical asymptotes. A vertical asymptote occurs when the function values approach plus or minus infinity on either side of a fixed point. This usually happens when a division by zero takes place in the function computation. Such transcendental functions as TAN(X) also have vertical asymptotes. The mechanism for handling these asymptotes is found in lines 180 and 190. The detection of asymptotes is found on line 150 where A is set to (M<(OY-Y)). A will be 1 if there is an asymptote and 0 otherwise. The determination is made by comparing the difference in the old function value and the latest function value. If that difference is greater than .7 times the height of the window, A is set to 1. The value .7 is arbitrary and can be adjusted if it does not pick up the asymptotes in a given function.

Advanced Graphing Techniques

There are several advanced graphing techniques to be considered. Techniques can include software that produce characters to mix with graphics, speed up loading from the disk or move blocks of memory, as well as special ways to do things related to graphics. The remainder of this article will look at several of these techniques.

Parallel Processing

Parallel processing implies that the computer is capable of doing two jobs at once. The advantage of parallel processing is that a job could be done in half the time if we could divide it between two processors. This is desirable but not possible on the Apple. However, there are two distinct jobs that must be done by interactive graphics programs. One of these jobs takes a lot of CPU time but no user time, the other takes a lot of user time but not much CPU time. The first job is that of computing the several hundred function values used in plotting the graph. The second job is collecting the data required to determine the domain, range, resolution, whether or not coordinate axes should be drawn, the distance between tic marks, and so on. This takes the user of the program some time to enter but lets the CPU idle most of the time. The secret in speeding up the entire graphing process is to overlap these two jobs as much as possible.

As soon as the resolution and the domain have been entered, the processor can compute all of the function values. There are still things that need to be entered in most graphing situations. While we are entering such values as the range or the distance between tic marks, the processor can be using its spare time to compute function values. Often it can compute all function values before the user has entered all of the particulars for the graph. When this happens, the only time required to plot the graph is that used to scale and draw the point.

If the function values are to be computed before they are plotted, they must be stored in a subscripted variable. This is easily done as follows:

```
DIM P(279)
```

The number 279 is chosen since that is the largest X coordinate that can address a pixel on the Apple screen. Instead of scaling and graphing each point as we did in the graphing algorithm given above, we will store each function value as it is computed in the array P. We need three subroutines to compute function values while we wait for input from the keyboard.

A subroutine to compute function values. This subroutine is responsible for computing one function value and incrementing all arguments. X is the world x-coordinate and XX is the corresponding screen coordinate. It is assumed that DX and the function FNF(X) have been previously defined.

```
300 REM COMPUTE AND SAVE ONE FUNCTION VALUE
305 IF X>RS THEN RETURN
310 P(XX)=FNF(X)
320 X = X + DX
330 XX = XX + 1
340 RETURN
```

Notice that this subroutine will not compute any additional function values once X exceeds RS, the right edge of the window.

A subroutine to start computing function values. This subroutine is needed to compute DX, set X to LS, XX to LM, and compute one function value.

```
400 REM COMPUTE FIRST FUNCTION VALUE
410 RES=RM-LM
420 DX=(RS-LS)/RES
430 X=LS
440 XX=LM
450 GOSUB 300
460 RETURN
```

A subroutine to 'scan' the keyboard. This is the subroutine with the two jobs. Its principle job is to accept input needed by the program. However, when it is waiting for the user to enter something (press a key), it can be calling subroutine 300. Subroutine 300 computes another function value every time it is called! This subroutine gets its input a character at a time. These are called GET loops and are much superior to the BASIC INPUT statement. Suppose we are asked to enter a number from the keyboard. The following GET loop will do the job and compute function values at the same time.

```
500 REM GET A NUMBER WHILE SCANNING
510 A$=""
520 POKE -16368,0
530 A = PEEK(-16384)
540 IF A<128 THEN GOSUB 300:GOTO 530
550 A = A - 128
560 IF A=13 THEN PRINT:RETURN
570 IF ((A>47) AND (A<58)) OR
    (A=43) OR (A=45) OR (A=46)
    THEN PRINT CHR$(A);:
    A$=A$ + CHR$(A): GOTO520
580 GOTO 520
```

When the <RETURN> key is pressed the input is found in the string A$. The term GET loop is used even though the GET function is not present. The reason is the similarity between a routine using the GET statement and the scanning routine given above. Note that line 570 will only add another character to the input string A$ when A has a certain value. This statement will screen out any character that could not be found in a number. When A=13, the <RETURN> key has been pressed. If you wanted the user to enter a 'Y' or 'N' in response to some question you could change line 570 to the following:

```
570 IF (A=78) OR (A=89) THEN
    A$=CHR$(A): PRINT A$: RETURN
```

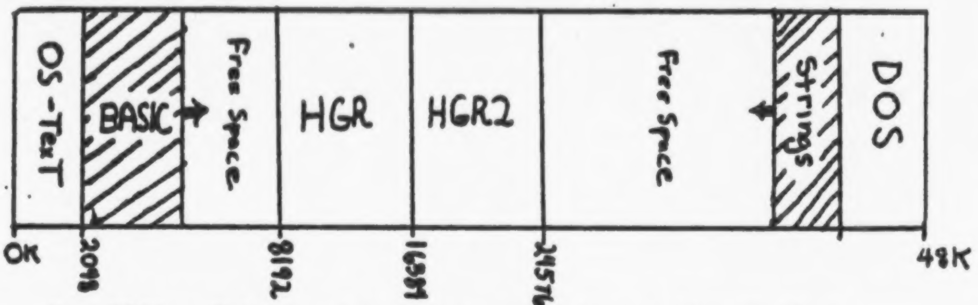
A routine that accepted answers A$ containing alphabetic characters would have a different line 570. In addition to the lines given above, a GET loop would probably have some lines for editing the string A$ in the case that an error was made. It is easy to remove the last character of A$ when A=8 (<BACKSPACE> key). Some changes would have to be made in the graphing algorithm above in order to use the array P to provide the function values.

Memory Allocation

Once you get beyond the trivial stage in graphics programming on the Apple you discover the graphics bit map. This will not be a pleasant encounter in most cases. Typically you will have spent several hours writing your program and just before you save it for the final time, you RUN it. The RUN will start out quite normally but suddenly the program will drop into the Monitor (machine language) or the program will freeze. When you finally regain control, you will discover that the end of your program has vanished! Of course the best ideas are always part of the missing code. What has happened is your program has become so large that it extended into the graphics bit map. When your program issued the HGR command, it erased that part in the bit map. Programs that exceed 6 kilobytes will collide with the bit map.

Figure 3, below, is called a memory map for the Apple. There are two graphics pages in this map, HGR and HGR2. BASIC programs will start at address 2048 and grow towards HGR unless redirected. There are several solutions to this collision problem. Two will be presented here. Others can be found in Roy Myer's text.

Figure 3:



Use HGR2. This is the easiest solution to the problem. Don't use the first page of graphics. This will give you 8 more kilobytes for your program to bring the total to 14 kilobytes. All you need to do is enter HGR2 wherever you previously used HGR. There is one slight hitch. HGR2 gives you full page graphics and you loose the four lines of text at the bottom. It is possible to regain those four lines using the 'soft' switches but not very pratical. A better procedure would be to purchase a shape alphabet or a character generator. With either of these you could place text anywhere on the graphics page. You would never miss the four lost lines of text! A shape alphabet will be discussed later.

Move BASIC to HGR2. This takes a little more doing than the first suggestion but it allows you to continue using HGR. Essentially, you direct the program to load at address 16384, the first location in the second graphics bit map. By doing this you discard the 6 kilobytes of memory below the first graphics page. However, this memory can be used for data storage, machine level programs, or storing shapes. Starting at location 16384 will give your graphics program about 24 kilobytes of space. The exact amount will depend on how many variables you are using. To load at this address, type the following commands before LOADing or RUNing your program.

```
POKE 104,64
POKE 103,0
POKE 16384,0
```

If you want to run your graphics program without typing these preliminary commands, you can place them in a program that does it for you. Suppose that your graphics program is stored on the disk as GRAPH. Type the following program and store it as RUNGRAPH.

```
10 POKE 104,0: POKE 103,0: POKE 16384,0
20 PRINT CHR$(4);"RUN GRAPH"
```

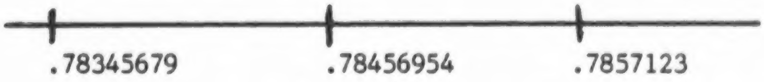
To RUN your graphics program and have it load at 16384, type:

```
RUN RUNGRAPH
```

No preliminary POKEs will be required.

Printing Numbers

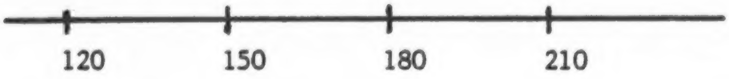
When labeling coordinate axes you will discover that the Apple will not always choose a pleasing decimal form for the coordinates. This is particularly true if you allow the computer to choose the distance between tic marks. You might have the following:



You really want enough decimal places to distinguish the coordinates but not so many that the axis is filled with numbers. The Apple will return numbers with as many as 15 digits. You can control this with the following subroutine. There are actually two routines here, one is called GOSUB 800 and the other GOSUB 810. UX and UY represent the distance between five tic marks evenly spaced on the X and the Y axis respectively. Each will contain just enough decimals to make the tic marks distinguishable.

```
800 AA=(RS-LS)/5: GOSUB 820: UX=AA: RETURN
810 AA=(TS-BS)/5: GOSUB 820: UY=AA: RETURN
820 T%=SGN(AA): AA=ABS(AA): AB=INT(LOG(AA)/LOG(10))
830 P=1: IF AB<>0 THEN
FOR J=1 TO ABS(AB): P=10*P: NEXT J:
IF AB<0 THEN P=1/P
840 AA= T% * INT(AA/P + .5) * P
850 RETURN
```

With RS=.0002 and LS=.00005, a GOSUB 800 would produce AA=4E-05. Notice that the mantissa is a single digit. If RS=234 and LS=112 then AA=30. The following tic marks could then be generated:



Using Shapes

One of the easiest ways to mix graphics and text on the Apple is to use alphabetic shapes. Shapes are binary codes that direct the Apple to draw special patterns. You must obtain a shape table containing the letters of the alphabet and then the table must be loaded into the Apple's memory. Shape tables may be created by the user or purchased from software companies. A program that creates a shape table alphabet can be found in Roy Myer's text. It is much easier to purchase a table or find one in the public domain than it is to create one!

Suppose that you have obtained a shape table called ALPHABET. It will be stored as a binary file on the disk. In order to use it, the table must be loaded into memory. The question is, where to put it? If you have moved your BASIC program above HGR (at 16384), then the space usually occupied by BASIC is available. The table could thus be loaded at 2048. On the other hand, if you are using HGR2 and BASIC has not been moved, then load above HGR2 (at 24576). Assume that you have moved BASIC above HGR and want the shapes loaded at 2048. The following command in your BASIC program will load the shape table as required.

```
PRINT CHR$(4);"BLOAD ALPHABET,A2048"
```

The next thing to be done is to tell the Apple that you want to use this shape table. This is accomplished by two POKEs. The numbers POKEd represent the address where you placed the shape table. The address is placed into the computer in two parts. Part one is the quotient when 2048 is divided by 256 and part two is the remainder. Thus part one = 8 and part two = 0. The necessary POKEs are given below.

```
POKE 233,8: POKE 232,0
SCALE=1: ROT=0
```

The SCALE and ROT commands are used to insure that the characters will be drawn with the right size and orientation. The following algorithm will print labels stored in A$ starting at location H,V. This algorithm assumes that alphabetic shapes are accessed by their ASCII code.

```
900 FOR J = 1 TO LEN(A$)
910 IF H>273 THEN 930
920 DRAW ASC(MID$(A$,J,1)) AT H,V:H =H+6
930 NEXT J
940 RETURN
```

Suppose that we want to print "SINE FUNCTION:" in the upper left corner of the screen. The following command could be used.

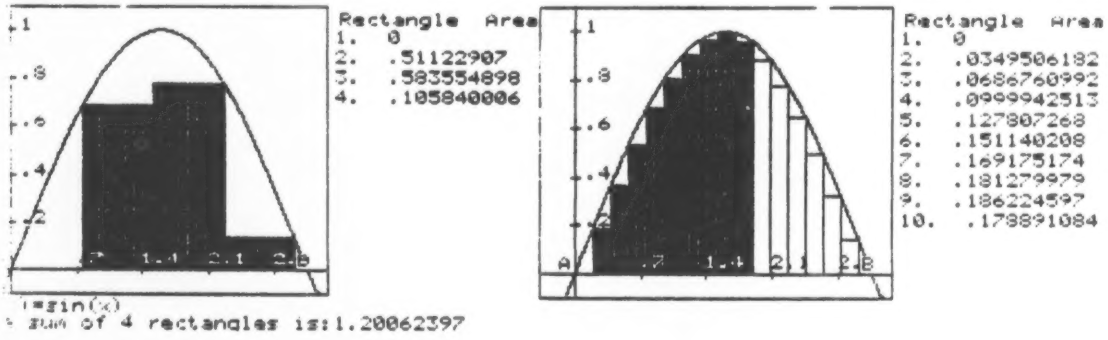
```
A$="SINE FUNCTION": H=10: V=10: GOSUB 900
```

If we want to print numbers using shapes then the numbers must be converted to strings first. This is easily done using A$=STR$(X) function. X is a number and it is converted to the string A$.

Moving the Graphics Page

When using graphs of functions to illustrate advanced mathematical concepts, you often want to recall the graph several times. That is, you create the graph, store a copy in memory, alter or add to the graph appearing on the screen and then recall the original graph which erases the additions. Suppose that we are illustrating the definite integral by summing the areas of rectangles inscribed under the graph of a function. We start with four rectangles, then 8 rectangles and eventually get to 32 rectangles. Each time we have the computer find the sum of the areas of the inscribed rectangles. This is illustrated in figure 4 below.

Figure 4:



```
Rectangle  Area
1.   0
2.   .51122907
3.   .583554898
4.   .105840006
```

```
=sin(x)
sum of 4 rectangles is:1.20062397
```

```
Rectangle  Area
1.   0
2.   .0349506182
3.   .0686760992
4.   .0999942513
5.   .127807268
6.   .151140208
7.   .169175174
8.   .181279979
9.   .186224597
10.   .178891084
```

Should we erase the scene when we go from one set of rectangles to another? No! If so, we would have to re-draw the axes, label them, and graph the function again. This would take time and eventually bore the user. The best solution is to save the graphics page in spare memory at the moment that it contains just the axes and function graph. We can recall it when ever we are ready to start a new rectangle drawing. If you have an Apple IIe or c, or a memory card in your Apple II plus, then you have ample room to store the graphics page above DOS (at 49152). The following program will load a machine language routine that will save and recall HGR in the area mentioned.

```
10 FOR J = 768 TO 843
20 READ X: POKE J,X
30 NEXT J
40 DATA 44,137,192,44,137,192,32,60
41 DATA 3,177,252,145,254,200,208,249
42 DATA 165,253,201,63,240,7,230,253
43 DATA 230,255,76,9,3,44,138,192
44 DATA 96,0,44,139,192,32,60,3
45 DATA 177,254,145,252,200,208,249,165
46 DATA 255,201,255,240,232,230,253,230
47 DATA 255,76,40,3,169,0,168,133
48 DATA 252,133,254,169,32,133,253,169
49 DATA 224,133,255,96
```

After this program is run, you can save the binary file that it creates on the disk. This binary file can then be loaded from the disk by your graphics program. The first line below will save the binary file and the second line can be used in any program to recall it.

```
BSAVE MOVE,A768,L80
PRINT CHR$(4);"BLOAD MOVE,A768"
```

Once this program is in memory, it can be used as follows. To save the current graphics scene use CALL 768. To recall from memory the scene that was saved last, use CALL 802. You will discover that the save and the recall are nearly instantaneous. It is so fast that it can almost be used for animation.

Summary

Graphics is a very powerful tool in mathematics. The ability to quickly graph, save and recall functions greatly contributes to this power. There are many useful algorithms that have not been included here. Almost as important as the graphing techniques is the user interface. That is, how the user interacts with the graphics program.

The algorithms given here are written for the Apple II family of computers. If you were to try to use the parallel processing subroutines or the shape table instructions on another computer, it would fail! The algorithms all reflect Applesoft BASIC and the Apple operating system. However, each computer has its own set of commands and these can be used to affect the same or similar results. The important thing is the concept. If you find some of these ideas useful, you should be able to implement them on almost any computer. Many of the ideas, as parallel processing and block memory moves, have applications out side of graphics as well.

Suggested Readings

Hill, Lee T. 'Problems in Computer Graphics.' Paper, The Computer: Extension of the Human Mind Annual Conference College of Education, Eugene Oregon, July 1983. 10 pages.

Myers, Roy E. 'Microcomputer Graphics.' Addison Wesley 1982. 281 pages.

Newman, William M. & Sproul, Robert F. 'Principles of Interactive Computer Graphics.' McGraw Hill, 1973. 607 pages.

Rogers, David F. & Adams, J. Alan 'Mathematical Elements for Computer Graphics.' McGraw Hill, 1976. 237 pages.

"Extending the Human Mind: Computers in Education"
July 31 - August 1, 1986

ABSTRACT

Presentation - Kathleeen M. Hurley

"The Impact of Optical Storage Technology on Education"

This presentation will focus on the new technology, CD-ROM, and its use for research, instruction, and administrative purposes.

CD-ROM stands for Compact Disc/Read Only Memory and is a small, 4 3/4" iridescent hard disc. This amazing technology has an enormous storage capacity of approximately 550 megabytes; enough storage capacity to put the name and address of every person in the United States on one side of a single disc. Unlike a floppy computer disc, a CD-ROM is an inexpensive circular piece of plastic that is reproduced in large quantities with pre-written information. The discs are inexpensive to manufacture, convenient to store, and virtually indestructable.

Unlike the floppy disc, the CD-ROM disc can be used in different manufacturers' compact disc drives so there is no problem of machine incompatability.

The primary difference between a CD-ROM and a laser videodisc is the way in which the signals are stored. The CD-ROM stores information in digital form and the laser videodisc stores data so that it fits the standard television

format. For example, Grolier Electronic Publishing has their 20 volume Academic American Encyclopedia available on both 12" laser videodisc and on CD-ROM. The encyclopedia takes up the entire side of a 12" laser videodisc, but only 20% of the CD-ROM.

CD-ROM will have an immediate and substantial impact on reference and research; it will take longer to work its way into the curricular software area but the impact there could be significant. In larger school systems CD-ROM could be used in managing the reams of policies and procedures needed in the administration of our school systems.

The most readily grasped educational use for CD-ROM lies in the area of reference materials--encyclopedia, periodical guide, dictionary, atlas, thesaurus, etc. Using reference materials on CD-ROM open new visions for students and promotes self-discovery.

Curricular software is another use of CD-ROM that holds promise. One CD-ROM could hold the equivalent of 1,500 IBM floppy disks or roughly 3,000 Apple disks. It would be possible to access thousands of software programs from a CD-ROM. Another possibility would be to have hundreds of levels of diagnostic and prescriptive test items tied to several thousand instructions items. This could work interactively with a standard magnetic floppy disk so that individual student progress could be recorded.

The other application for CD-ROM would be that of administration. School administrators must deal with reports, rules, procedures from federal, state, and local education agencies. It would be possible to receive a single CD-ROM every month or so that contained all of the documents needed - past and present - with a simple index or searching system. The district would reap savings benefits from not having to print, distribute, file, update and retrieve old print materials.

Costs of the hardware needed to use the CD-ROM are dropping which should influence administrators, librarians, and media coordinators to seriously start investigating CD-ROM for their schools and districts.

APPLICATIONS IN THE CLASSROOM: PRINT SHOP, PRINT SHOP COMPANION AND OTHER "EASY" PROGRAMS

by Brian James

Although it is steadily increasing, the actual percentage of computer using teachers and students still seems to be small considering the number of years educators familiar with computers have been spreading the good word. Up until very recently the obvious blame for this was the lack of sound educational software and the fact that application software was often complicated and time consuming to learn and use. Too often when learning and using a piece of software was compared to what was gained with it's use, many educators and students opted to finish the task without the use of the computer.

Beginning with the introduction of such programs as the Bank Street Writer and other "easy" word processing systems, the concept that application software is too hard to learn is quickly changing. These software products are relatively fast to learn and yield something that is actually saving time and improving the end result. Bank Street Writer has been a great way to introduce students and teachers to positive and worthwhile computer use quickly and easily. Most often it challenged the student and teacher to investigate other practical uses of computers.

Many other software products are coming to market using the "Bank Street" philosophy - easy and quick to learn, yielding improved results over what was done before. Even more encouraging is the increase of software using visual operating systems based often on the Macintosh icon system which makes learning an application almost intuitive. Many of these products initially were marketed for home use but have been quickly picked up by educators as great ways to improve both their own and their students productivity. Three of these packages will be highlighted here: "The Print Shop", "The Print Shop Companion", and "Dazzle Draw"

The Print Shop

The Print Shop is the oldest of the programs discussed here. Released in 1984 by Broderbund Software, The Print Shop is now available for the Apple // family, the IBM PC, Atari and the Commodore 64/128. A Macintosh version of the program is due to be released in the middle of August, 1986.

The Print Shop may be used to print professional looking greeting cards, signs, letterheads and banners. These are all things that students and teachers often spend time doing by hand. Unless a student or teacher is particularly artistic, results of their hand work is often unsatisfactory.

The "greeting card" option could be used for invitations, announcements, thank-you notes and any other "card" type activity. The teacher can either design a card style for the students to use or have each student design their own card.

Uses of the sign and banner options are numerous. Students and teachers may make signs with announcements and instructions. Students may use the sign option to make report covers. Banners, long signs printed on continuous fan-fold paper, can be used for dance or assembly announcements or in any other area where a large attention getting sign is needed.

The letterhead option provides a way to produce custom letterhead paper for a class or an individual student. The letterhead option is useful in assignments that have to do with letter writing or perhaps the foundation of a classroom or school newsletter.

The first page of The Print Shop manual states: "The Print Shop is so easy to use, you don't need to read this manual. Just start up the program and follow the prompts. You will be printing in minutes!" Unfortunately most users will take this invitation literally and pass up two of the more interesting and useful options of The Print Shop - Screen Magic and Graphic Editor.

The Screen Magic option allows teachers and students to load graphic images produced by The Print Shop or other compatible graphic programs and to add text. These screens may be printed on paper or saved on disk. The reverse side of The Print Shop master disk contains sample pictures that may be used or edited. Screen Magic also contains a "kaleidoscope" option that produces a series of patterns that may be frozen and used as background for a typed message.

The Graphic Editor contained in The Print Shop allows students and teachers to design their own pictures that may be saved and used later in a Print Shop sign, banner, greeting card or letterhead. Using this option, the school or class mascot may be incorporated in a Print Shop production or perhaps a signature may be used in a design. A keyboard, Koala Pad, or joystick may be used to help create the graphic image. The Graphic Editor may also be used to modify one of the Print Shop pictures that comes with the package or in one of the many graphic libraries now available.

The Print Shop is an excellent example of software that needs very little instruction and provides an improved method of doing something that had to be been done before by hand. The computer becomes a means to an end and not an end in itself.

The Print Shop Companion

The Print Shop Companion was released in September of 1985 as a supplement to the original Print Shop. The program can not stand alone – the user must also have a copy of The Print Shop. The Print Shop Companion operates with the same easy prompt menu as The Print Shop and many teachers and students can probably use many of its features without reading the instruction manual. The Print Shop Companion adds a number of fonts, pictures and borders for use with the original Print Shop. In addition, the Print Shop Companion allows the user to make their own fonts and borders using special "editors" contained in the program.

By far the most useful portion of The Print Shop Companion is the calendar maker. A teacher or student may make a monthly or weekly calendar that may be printed in small poster form. Teachers can easily make lesson plans or assignment calendars for their students. A monthly birthday calendar can be made to show class birthdays and other important dates. The calendars may be saved to floppy disk and revised when needed.

The Print Shop Companion also contains a "Creature Maker." Small children can mix and match body parts from a variety of crazy creatures and make their own characters. These characters may be printed as posters, greeting cards, calendars, banners, letterheads and signs. The creatures may also be used to form a coloring book or, using special ribbons, printed as iron-ons for T-Shirts. Using a backward font, available by flipping letters backwards in the font editor, text may be added to the T-Shirt. An entire class can easily and very cheaply manufacture their own T-Shirts set.

The Print Shop Companion adds a number of other features useful to students and teachers. Large number posters may be assembled using the graphics editor. The numbers may be used as signs or perhaps printed on T-Shirts for inexpensive team uniforms. Pictures from other programs, including Dazzle Draw, may be captured for use as Print Shop graphics. This means that more sophisticated designs made from other programs can now be used in posters and signs. Finally, The Print Shop Companion supports the use of a computer "mouse" allowing better hand control for art work.

Dazzle Draw

Dazzle Draw from Broderbund Software is essentially a drawing program. It is available only for an Apple //e with extended 80 column card or an Apple //c. The program is unique in that it uses "pull down" menus and tools similar to MacPaint for the Apple Macintosh. The program needs either a "mouse" or "Graphics Tablet". The "mouse" seems to come closest to an artist's manipulation of a pencil or brush.

The Dazzle Draw menu contains aids to making squares, lines, circles and a variety of drawing patterns and styles. Colors may be used on screen which are translated to shades when printed on a black and white printer (the program also supports color printers). The Dazzle Draw menu also contains an option for a limited number of text fonts, mirror and picture moving or repeating controls.

Although the obvious use of Dazzle Draw is for students and teachers to create pictures, because of the many menu options, the program should not be limited to free form drawing. The line and text options can be used to make tables needed to be included in reports (or just to learn about tables). Teachers may easily construct a graphic seat chart that may be changed and reprinted as students change. The circle, line, shade, and text options can make pie charts. The rectangle command will help students with bar graph assignments. As with The Print Shop, Dazzle Draw can help produce signs and letterheads. Dazzle Draw pictures may be moved to The Print Shop Companion. Just about any graphics assignment that may have been somewhat frustrating and time consuming by hand drawing will become easier and more precise with Dazzle Draw. Students are able to spend more time with the substance of their presentation instead of worrying how to draw a circle.

CONCLUSION

The Print Shop, The Print Shop Companion and Dazzle Draw all represent an exciting new generation in application software. They are important in that they produce a product not easily done by most students and teachers by hand. Students and teachers need to spend a minimum time to learn the programs. The substance of the exercise becomes more important than the fact that it happens to be computer aided. The computer is simply a tool to achieve an end easily and quickly. With more programs coming to market following this philosophy, we will certainly see a greater percentage of computer using teachers and students.

List of programs featured:

"The Print Shop" by David Balsam and Martin Kahn, produced by Broderbund Software, Inc. Apple // family, Commodore 64/128, IBM PC, Atari (8 bit machines), Macintosh version release in August, 1986.$49.95

"The Print Shop Companion" by Roland Gustafsson and others, produced by Broderbund Software, Inc. Apple // family, Commodore 64/128, Atari. $39.95

"Dazzle Draw" by David Snider, distributed by Broderbund Software, Inc.,Apple //e (128K), Apple //c $59.95

CREATING SCHOOL DOCUMENTS WITH PAGEMAKER

Julia A. H. King

The pen is mightier than the sword. E. G. Bulwer-Lytton

He loved his pen, they loved their swords--it's desktop publishing I adore. J. King

If you're one of the people designated by your school to create printed materials--newsletters, teaching aids, brochures, fliers, newspapers, policy manuals or handouts of any kind--you've probably already heard the term *desktop publishing*.

Desktop publishing methods enable one person, using a computer, to write, illustrate, layout, paste up and print entire publications. Until recently, this application required an expensive and cumbersome computer system. With the advent of the inexpensive laser printer and several good page make up programs, it is now possible for most school systems to own and use a desktop publishing system.

In this paper we will examine the use of a Macintosh computer, an Apple LaserWriter and Aldus PageMaker for the creation of school documents.

First of all, it's important to understand what PageMaker can do. It's a page make up program designed to allow you to use the Macintosh to lay out text, illustrations, and graphic elements (lines, borders and boxes) on a page. It replaces the more traditional tools of graphic design and paste up: the drawing board, T-square, ruling pens, typesetting machine, layout board and rubber cement. PageMaker does not, however, replace experience or expertise in graphic design. The program has no means for making aesthetic decisions..the beauty of the piece is still up to you. This program, combined with a Macintosh and a LaserWriter, simply provides a flexible medium for experimenting with page make up and printing. If you decide you need to change the date on a flier, PageMaker will let you make the change without having to pull glued-down type from a page or sending out for revisions from the typesetter.

Beginnning

When you have a document to create, the first step is to decide on a general design. My favorite way to do this is to look through my files and books to find examples of designs that I enjoy, and then to choose the kind of design that works best for this particular piece.

Next, I make a few little rough sketches, on scratch paper, of how the document might look. I ask myself questions such as:

- What is the purpose of this document?
- How many columns will this document have?
- Will this document have illustrations?
- What kinds of typefaces will I use?
- Will this document have borders?

and so forth.

Writing the Text

All the text is then prepared as *MacWrite* files. If the final document will be printed on a LaserWriter, I try to use the special LaserWriter fonts such as Times and Helvetica. I create each article and each headline as a separate *MacWrite* file. That way, text can be pasted, cut and moved around without affecting other blocks of text. This method takes a little extra time initially, but is much more convenient in the long run.

Preparing the Illustrations

The second task is to prepare the illustrations. *PageMaker* can accept illustrations created in *MacPaint*, *Mac Draw*, computer clip-art collections and digitized artwork.

I put each illustration in a separate file and try to store the art in the size that it will appear in the publication. It *is* possible in PageMaker to scale graphics up or down and to "crop" artwork, but these tools are better used for minor adjustments rather than large scale changes.

Creating a Template

The next step in designing the document is to create a *PageMaker* template. This template will show the blank pages, in their proper format, for laying out the document. When creating a template, I ask myself questions like these:

- How many pages will this document have?
- How many columns will each page have?
- How will text and graphics be integrated?
- How big will the borders be?

Placing the Elements

Now that the template is in order, you may begin to place the text and graphics on the page. The first items to be place are the outside borders, if there are any, and then the major headings or other graphic elements (the masthead, in a newspaper, for example). Then, I place the illustrations, and arrange the text around them.

Concluding

Once the text and illustrations are in place, minor adjustments may be made, font sizes may be adjusted, and illustrations may be bordered and cropped. Final embellishments like thin rules between columns, page numbers, date lines and a few decorative elements can be added.

Once the design is complete, it must be saved to the disk. The final design is them printed.

The first printing will nearly always reveal minor changes that need to be made. There will often be some slight discrepancy between what the screen displays and the appearance of the printed page. Spelling errors will magically appear and a few things will probably need subtle repositioning.

A final copy is then printed and then delivered as camera-ready copy to the printer, or photocopied. The LaserWriter can be used for very small editions, but Apple Computer suggests that the LaserWriter not be used for more than 3000 pages per month, so it has limited use as a printing press.

The ability to control the entire process of producing a printed piece will allow busy desktop publishers to cut down their turnaround time, and increase their production without increasing their work load. Even better, it will create the wonderful sense of achievement of having done a job yourself and having done it well.

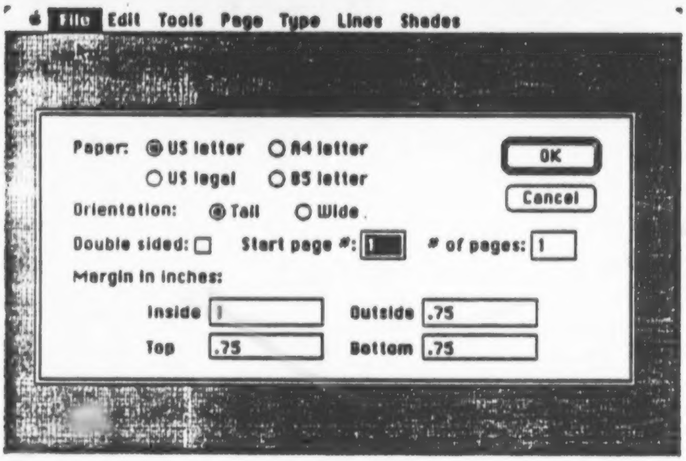
• The following handouts are reprinted from August 1985 ShowPage, Volume 2, # 7. The article was written and permission was granted to reprint by David Healy, First Image, San Francisco, (415) 864-7033.

PageMaker Tour

A GRAPHIC REVIEW BY DAVID HEALY

This article was composed entirely in PageMaker.

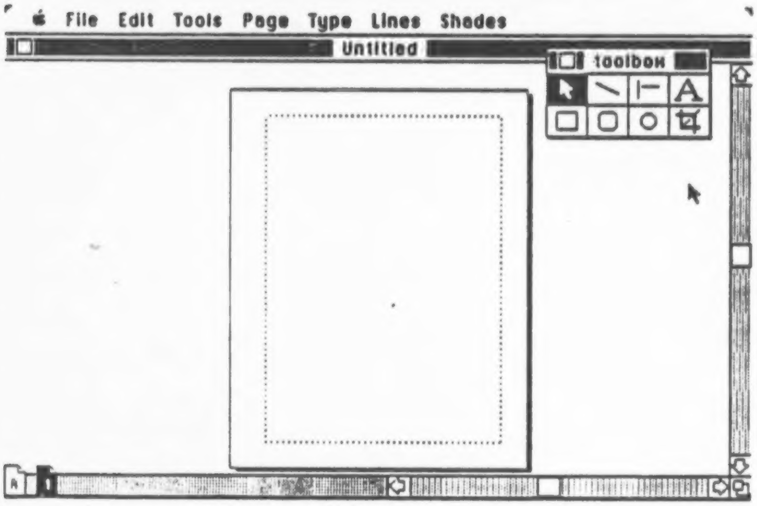
When you first open the PageMaker application you will see the ***Page setup*** screen that gives you choices about the: size of the page, the orientation of the page, the number of pages (up to 16 pages), whether it is single or double sided and the margins widths.

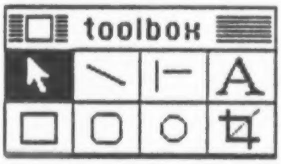

File Edit Tools Page Type Lines Shades
OK
Cancel

Paper: ● US letter ○ A4 letter
○ US legal ○ B5 letter
Orientation: ● Tall ○ Wide
Double sided: ☐ Start page #: 1 # of pages: 1
Margin in inches:
Inside 1 Outside .75
Top .75 Bottom .75

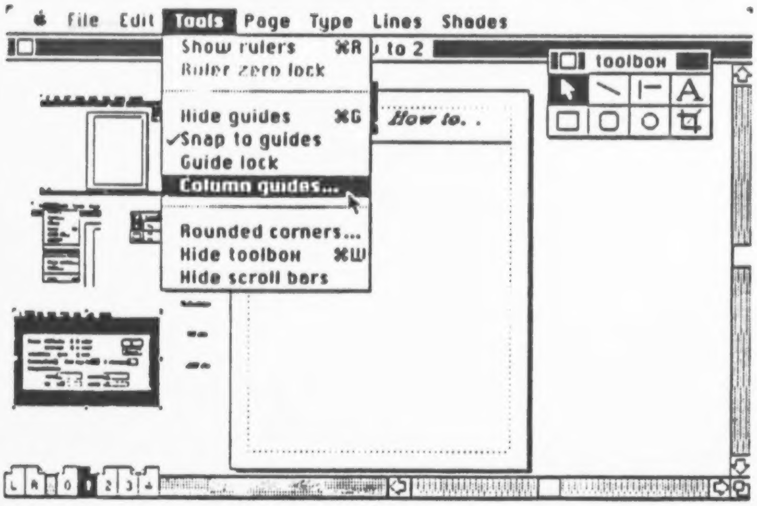
After completing the specs a blank page will appear on the screen.

The blank page can be seen at left on the pasteboard. Elements may be placed on this pasteboard while you figure out how you want to use them. If you flip between pages the pasteboard remains the same allowing you to easily play with elements on number of different pages.

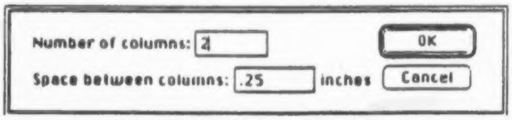

File Edit Tools Page Type Lines Shades
Untitled
toolbox
A



One of the first things you will notice is the **toolbox.** Going from left to right, first is the **arrow** which enables you to activate any graphic elements for positioning, proportioning, cropping among other activities. **Rubber band** line and **vertical & horizontal lines** are next. At the right is **A** for entering or altering text. The **box, rounded box,** and **circle** are just like the MacPaint or MacDraw functions. The tool in the bottom right is for cropping MacPaint or MacDraw documents.

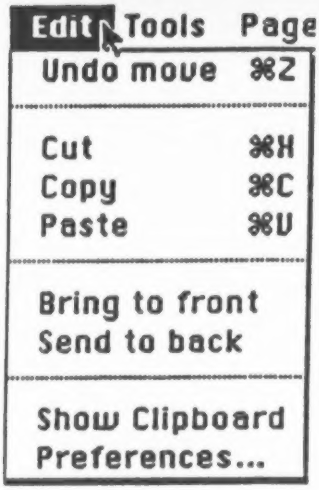

File Edit Tools Page Type Lines Shades
Show rulers ⌘R
Ruler zero lock
Hide guides ⌘G
✓Snap to guides
Guide lock
Column guides...
Rounded corners...
Hide toolbox ⌘W
Hide scroll bars
to 2
toolbox
A
How to. .

For the placement of text **Column Guides** may be chosen from the **tools** menu. A box will appear requesting the number of columns and the space between them. You may have up to 10 columns.



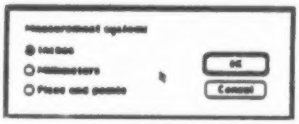
Number of columns: 2
Space between columns: .25 inches

BEST COPY AVAILABLE

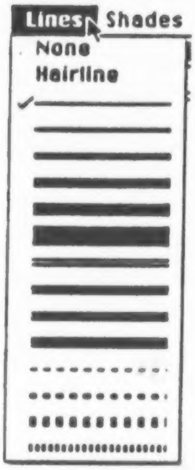

Edit
Tools
Page
Undo move ⌘Z
Cut ⌘X
Copy ⌘C
Paste ⌘V
Bring to front
Send to back
Show Clipboard
Preferences...

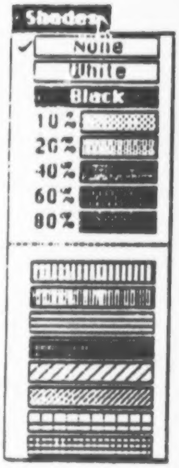
Most of Edit commands are very much like those in most other Mac-Design programs we are used to.

Preferences... is a feature for deciding what unit of measure you prefer.



A number of other graphic options, very much like those in MacPaint and MacDraw are available.


Lines
Shades
None
Hairline

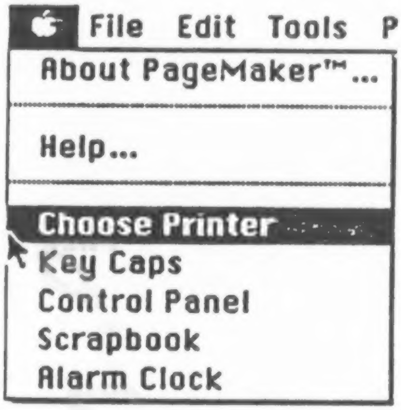

Shades
None
White
Black
10%
20%
40%
60%
80%

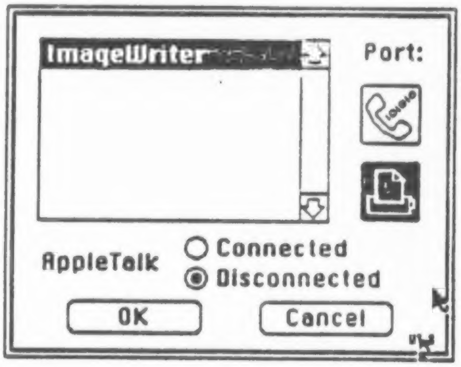
For printout you have choices depending on what is connected to your Mac. When you **Choose Printer** a box appears like the one at the far right.

When it the LaserWriter is connected and choosen, its command box appears.

This gives you a choice of the number of copies, collated, the particular pages from your file, and scaling among other features.

The scaling feature lets you choose any size from a 15% reduction to a 1000% enlargement. It also will print out a thumbnail where all of the pages in a particular file will appear reduced on one page.

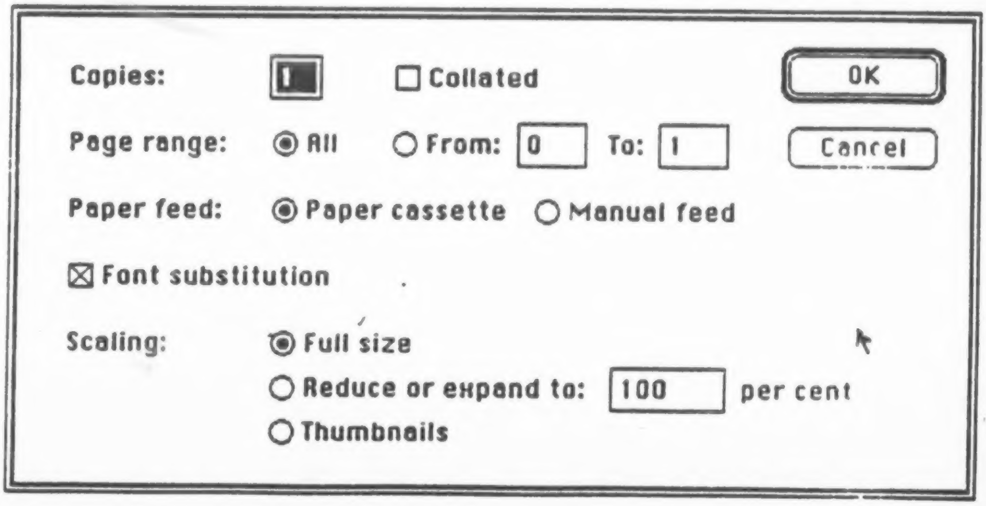

File
Edit
Tools
P
About PageMaker™...
Help...
Choose Printer
Key Caps
Control Panel
Scrapbook
Alarm Clock



ImageWriter

Port:

AppleTalk ○ Connected ● Disconnected



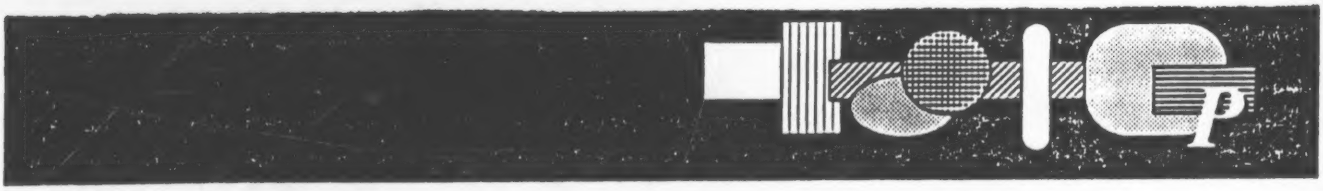
Copies: 1 ☐ Collated

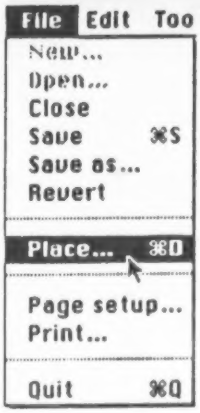
Page range: ● All ○ From: 0 To: 1

Paper feed: ● Paper cassette ○ Manual feed

☒ Font substitution

Scaling: ● Full size
○ Reduce or expand to: 100 per cent
○ Thumbnails

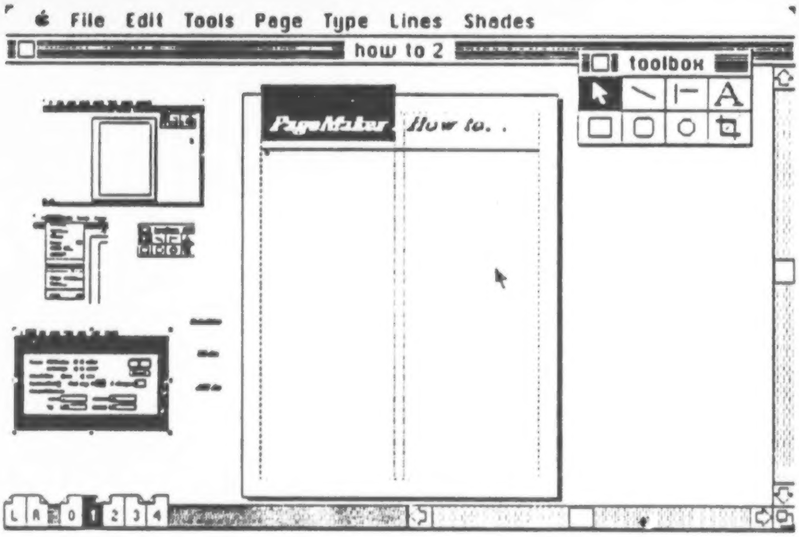


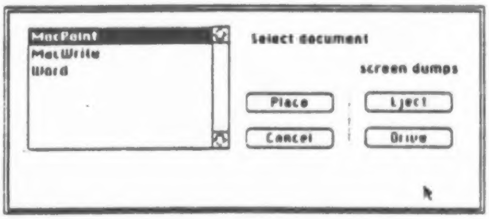

File
Edit
Too
New...
Open...
Close
Save ⌘S
Save as...
Revert
Place... ⌘D
Page setup...
Print...
Quit ⌘Q

At right is the page in fit in window size with two columns established.

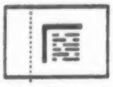
Documents from MacPaint, MacDraw, MacWrite, and Word may be placed into PageMaker by activating the **Place** command.

A window will appear from which you may choose the particular document you want.

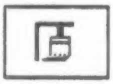

File
Edit
Tools
Page
Type
Lines
Shades
how to 2
toolbox
PageMaker
How to. .
L
R
0
1
2
3
4


MacPaint
MacWrite
Word
Select document
screen dumps
Place
Eject
Cancel
Drive

Depending upon which application you choose to place you wil get a corresponding icon:

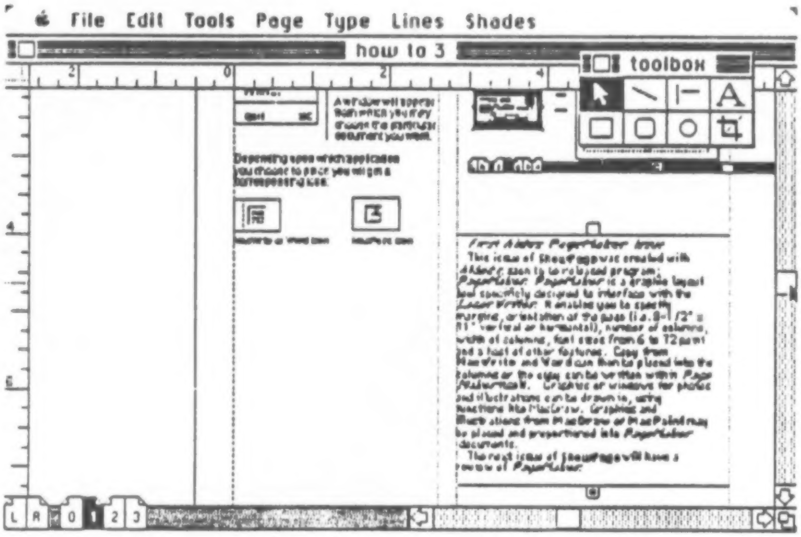


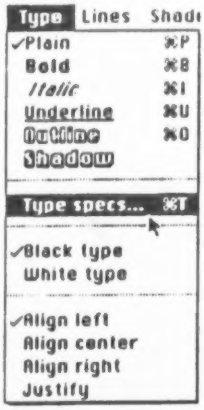
MacWrite Icon



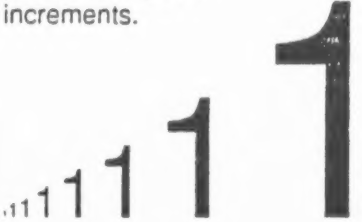
MacPaint Icon

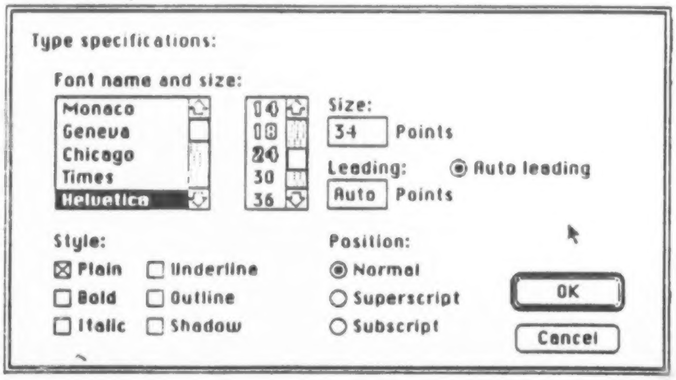
For text, once it has been placed it may be altered. You may change the: font, it's size, the line spacing, and the style.


File
Edit
Tools
Page
Type
Lines
Shades
how to 3
toolbox
First Aldus PageMaker Issue
L
R
0
1
2
3


Type
Lines
Shade
✓Plain ⌘P
Bold ⌘B
Italic ⌘I
Underline ⌘U
Outline ⌘O
Shadow
Type specs... ⌘T
✓Black type
White type
✓Align left
Align center
Align right
Justify

The sizes listed are the most commonly used. You may also enter your own size from 4 point to 127 point in one point increments.





Type specifications:

Font name and size:

Size: 34 Points

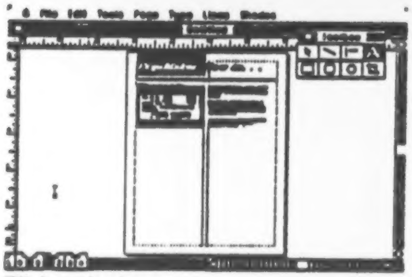
Leading: Auto Points ● Auto leading

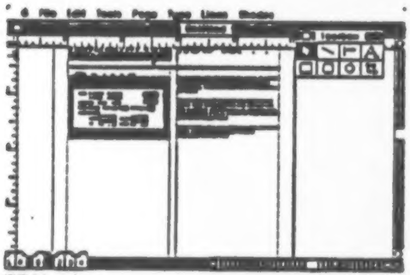
Style: ☒ Plain ☐ Bold ☐ Italic ☐ Underline ☐ Outline ☐ Shadow

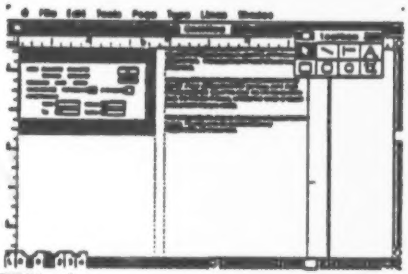
Position: ● Normal ○ Superscript ○ Subscript

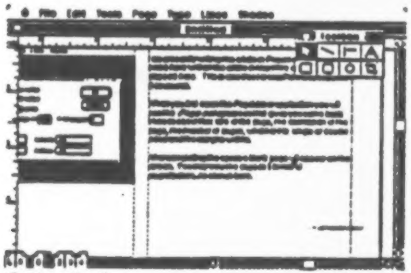
Monaco
Geneva
Chicago
Times
Helvetica
10
18
24
30
36
OK
Cancel

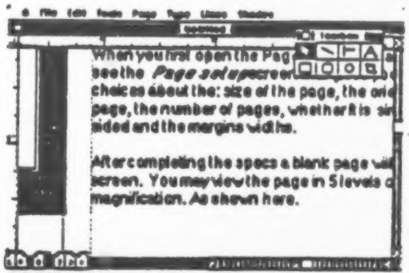
PageMaker Tour cont.


Fit in window


50% size

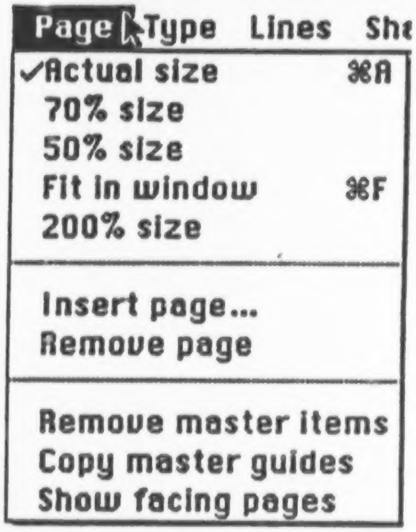

70% size


Actual size


When you first open the Pag
see the *Page setup* screen
choices about the: size of the page, the orie
page, the number of pages, whether it is sin
sided and the margins widths.

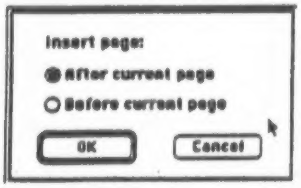
After completing the specs a blank page wil
screen. You may view the page in 5 levels o
magnification. As shown here.

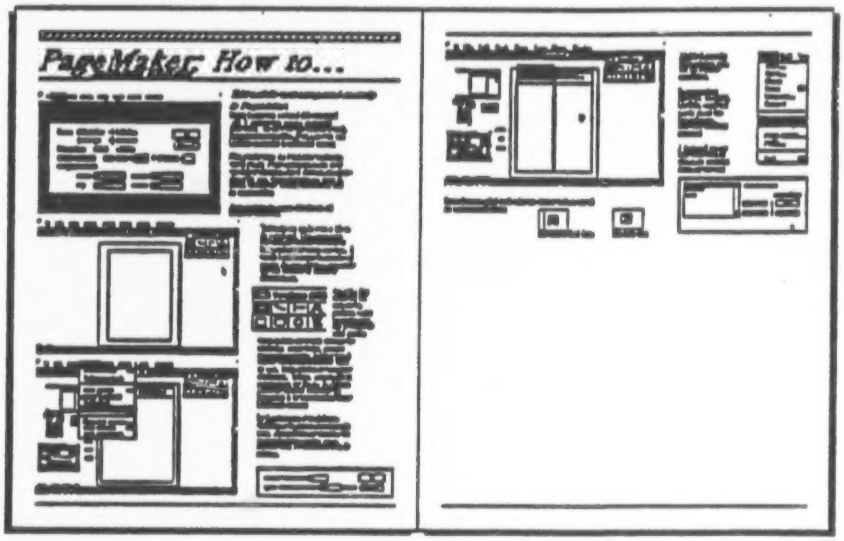
200% size


Page
Type
Lines
Sh
✓Actual size ⌘A
70% size
50% size
Fit in window ⌘F
200% size
Insert page...
Remove page
Remove master items
Copy master guides
Show facing pages

The page may be viewed at 5 different sizes as shown at left. Viewing at **Actual size** is the most accurate because, as you may notice, there is some variation in line length between the sizes.

Pages may also be deleted or new ones may be added, up to a limit of 16 pages.


Insert page:
● After current page
○ Before current page
OK
Cancel


PageMaker: How to...

Show facing pages is a feature that enables you to view a two page spread at once

BEST COPY AVAILABLE

NEW DATABASES IN THE SECONDARY CONTENT AREAS

TOM LAYTON SOUTH EUGENE HIGH SCHOOL

SEHS began its serious entry into educational computing by first placing microcomputers in the hands of teachers rather than in the hands of students. We reasoned that, if a school has extremely limited resources for the use of computers in schools,then education in general could be better served by making it possible for teachers to do their normal work in less time and with less effort. So, we all learned to use **Print Shop** to make signs, **Crossword Magic** to make puzzles, **The Report Card** to keep our grades, and **AppleWorks** for student records and the preparation of handouts. As expected, most of the faculty got tired of dragging Apples home on weekends and back on mondays and bought their own home computer. Unexpectedly, teachers didn't automatically translate their understanding of how to use computers for themselves into a sense of how to use computers with students.

Of course, by this time,the math department was using some computers to teach BASIC and PASCAL programming; and the business department began to integrate dedicated word processors and microcomputers into some classes. We even offered a class or so to teach "computer literacy" -- whatever that was. For the most part, however, we were not using computers to drive content. We purchased some *canned* educational programs to try out. Some were games and simulations, but most would be classified as Computer Assisted Instruction (CAI). Generally, the very few programs that existed were expensive, seemed poorly designed, never quite fit into the existing curriculum, and quickly bored the students. Their only advantage was that the teacher using them needed only the most rudimentary knowledge of computers. For a faculty already using computers on a daily basis, that was not much of an advantage. It was nearly useless to try to convince teachers to use computers in their classes simply to expose students to the kind of technology those students might encounter after graduation. Teachers rightfully felt that their job was to deal with content, and anything that did not help them deliver the kind of education they were hired to provide, only competed for precious contact time.

Those of us who moaned and groaned about the lack of good educational software did not realize that it was right there in front of us all the time. We did not seem to understand that word processors, databases, and spreadsheets were the most powerful educational tools we could imagine. We are just now starting to understand how these tools can be used to carry the content we have so desperately been trying to present in our classrooms. The difference between using computer tools and CAI is that teachers must not only know how to use these tools, but how to teach students to use them in ways that enhance an understanding of the subject under study.

It was the English department that really began to integrate computers into regular classroom work. Word processing and composition seemed to go hand in hand and provided immediate and obvious benefits. Spelling checkers, grammar checkers, and outline processors were natural additions to those efforts. Despite a serious lack of hardware, imaginative scheduling and financing guaranteed that all 10th grade students were able to experience the benefits of writing with a computer.

Word processing worked well for the English department, but where could we go from there? During the 1984-85 school year, District 4J offered teachers the opportunity to get together and try to look at using AppleWorks to create word processing, database, and spreadsheet files that might be used in the classroom. Twenty or so teachers took their first furtive steps toward computerizing some part of their individual classroom work. Undoubtedly, the stars of the show were the social studies teachers. They found a natural ally in data base programs.

Bob Veeck and Alice Jagger, with the help of Charlie Shoup and June Reams, created THE POWER OF NATIONS database working under the AppleWorks format. The data base consists of two files -- "Power" and "Special". The "Power" file (26k) contains 29 categories of statistical data on every nation-state in the world. The "Special" file contains 26 categories (some repeats from the "Power" file) on the same 167 nation-states and takes up 52k of the normal 55k AppleWorks desktop limitation. Samples of the two files accompany this article together with a portion of the student glossary they prepared in an AppleWorks word processing file.

Within the last two school years, THE POWER OF NATIONS has been used in various ways with five of the social studies instructors at SEHS and the International High School. At first, some teachers used the data base simply as a means of preparing handouts for their classes. They used the Apple IIe in the social studies office to prepare for class by arranging and printing the particular data they needed for that day's lesson. The handouts were reproduced for each student. This method was highly successful. Lesson preparation time was greatly reduced and the tables and charts of data were always exactly the way the teacher wanted them to be. However, as the students began to work with the data in the classroom, it became frustrating that the information remained fixed upon the page. It could not be manipulated, nor could data that the teacher had not prepared for the day be examined. For example, if the lists of data were arranged alphabetically by country name, it was difficult for the students to see which five countries in the world have the highest PCI or GNP; and whether they were the same five countries.

It quickly became obvious that it was necessary for the class to manipulate the data in class the same way the teacher did in her office. An Electrahome video projector was brought into the classroom along with an Apple IIe and THE POWER OF NATIONS data base. The projector allowed a whole classroom to view the data as it was being manipulated on the computer. The teacher could then respond to class discussion by arranging the data on the screen that was pertinent to the discussion at hand. If the discussion led one class in one direction and another class in another direction, appropriate data could be instantly reviewed.

As soon as students saw the power of computer manipulated data in their classroom, they wanted to be able to use the computer themselves to explore their own individual interests. They saw the great advantages over spending hours in the library digging through reference material, as well as the ability to view the information from many different angles. Those students who had learned AppleWorks in other classes asked for access to the data base and it was decided to bring five computers into the classroom and let one experienced student push the buttons on the computer for a group of five or six other students gathered around the computer screen. This method was just as successful as the previous two. Soon more students wanted to manipulate the machines themselves. It was arranged to bring the entire class to the Computing Center for a week of database work.

The culminating activity for all Global Studies classes for the year was a real time simulation of the nation-states of the world. Each student (or small group of students) took the part of an individual country. This simulation was a traditional class activity for South students, and they enjoyed coming dressed in native costume and making trading agreements, starting conflicts, and gaining or losing world power. Unlike previous years, however, students had spent many hours in the Computing Center preparing for the event by manipulating the data bases they had worked on during class time. Students knew, for instance, that their country needed manufactured goods and had lots of sugar. So they spent time looking for countries that needed what they had and had what they wanted. Friendly countries could be identified by similar languages, religions, and ethnic groups. For the first time, computers were available during the actual simulation so that data could be gathered and analyzed on the spot, and proof could be offered to judges that various actions were indeed supported by the facts. Students appreciated having computers available; and those students whose classes did not use databases during the year were at an obvious disadvantage.

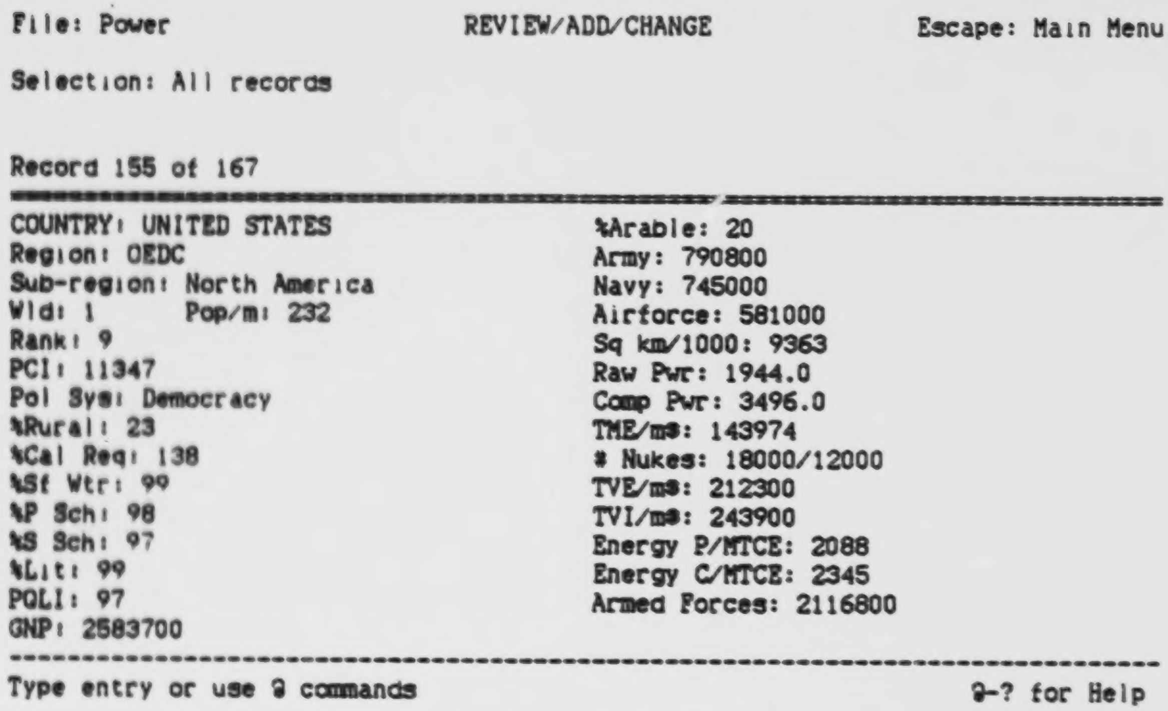
The unquestionable success of THE POWER OF NATIONS database has encouraged our teachers to seek new opportunities to use databases in their classrooms. Three basic avenues for database use are open to us. The first is to continue what we have already done -- create more databases to match the curriculum currently in place. The disadvantage, of course, is that creating a really good database for student use is quite time consuming. When asked how long it took to do THE POWER OF NATIONS, Veeck, Jagger and company refused to even estimate the long hours spent updating and revising the material. However, all contend that it was worth the effort.

Another avenue for teachers who wish to use databases in the classroom is to obtain them commercially. The 1986-87 year will probably become the year of the database. Two databases published by Scholastic, Inc. (running under PFS) in American History and US Government have been available for about a year. NewsWorks and NewsWorks II (running under AppleWorks) is published by Newsweek Magazine and comes free with a classroom subscription to their magazine. It is rumored that there will be as many as 25 databases commercially available by January 1987 from several different major software publishing houses. These databases will be aimed at the K-12 market and run under various batabase programs. It is also rumored that the databases will not be restricted to social studies. Two prominent releases will be offered for English and Science.

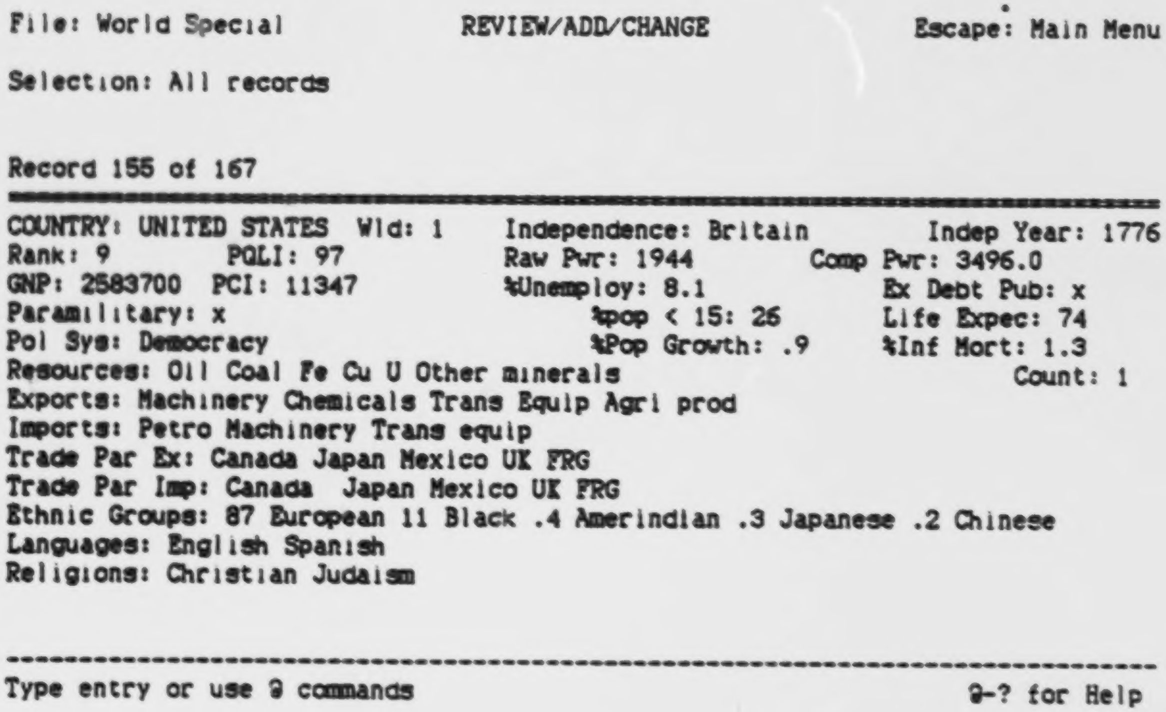
The last avenue for teachers to explore the use of databases to drive content will be in the creation of databases by students as a classroom activity. This is a very different way of using databases and involves some very different set of skills on the part of the student and the part of the teacher. Many teachers have experienced failure with this approach because they have asked their students to build a database before the students have actually used one. For beginning databasers, it will probably be best for the teacher to create the categories in the database and have the students do most of the research. More experienced students will enjoy creating their databases from scratch. Students will learn a great deal about the nature of data and databases by building their own. Other skills also come into play as students realize the real importance of correct spelling and careful information entry.

Before SEHS teachers discovered databases they were mostly disappointed in educational software. A few good packages existed, but they were slow in coming onto the educational scene; they were expensive; they were short lived in the classroom setting; and their educational focus was been fuzzy at best. Perhaps their greatest shortcoming, however, was that they tended to separate students from their classmates and teachers rather than bring them together.

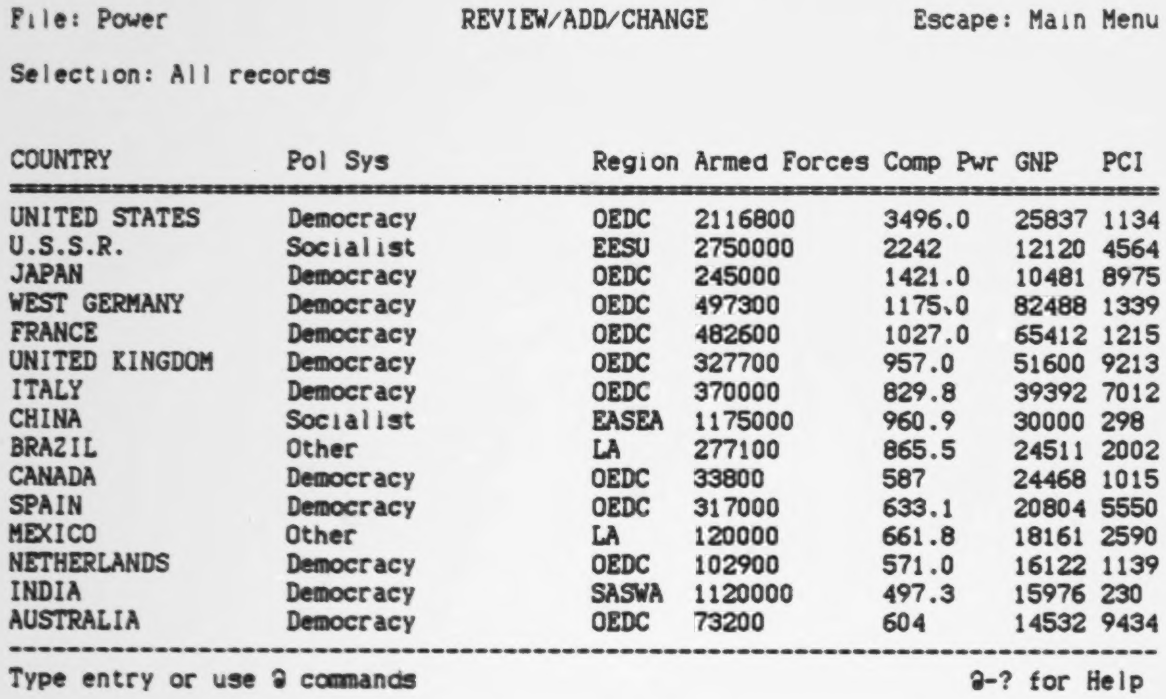
After our initial exploration with computers, students, and databases, it has become excitingly obvious that data, inspired by classroom content and manipulated by data base software, can bring teachers and students together on a grand adventure of intellectual exploration. Appropriate data can be researched, assembled, and incorporated into a computer environment very quickly by people who are intimately involved with students in content areas but who have no programming background. The databases themselves can become the basis for an entire year's worth of computerized lessons, and each one can spark even more lessons/discussions away from the machines. Students can feel a sense of mastery, and teachers can feel more involved in the process with the hundreds of new questions that the databases inspire in their students.



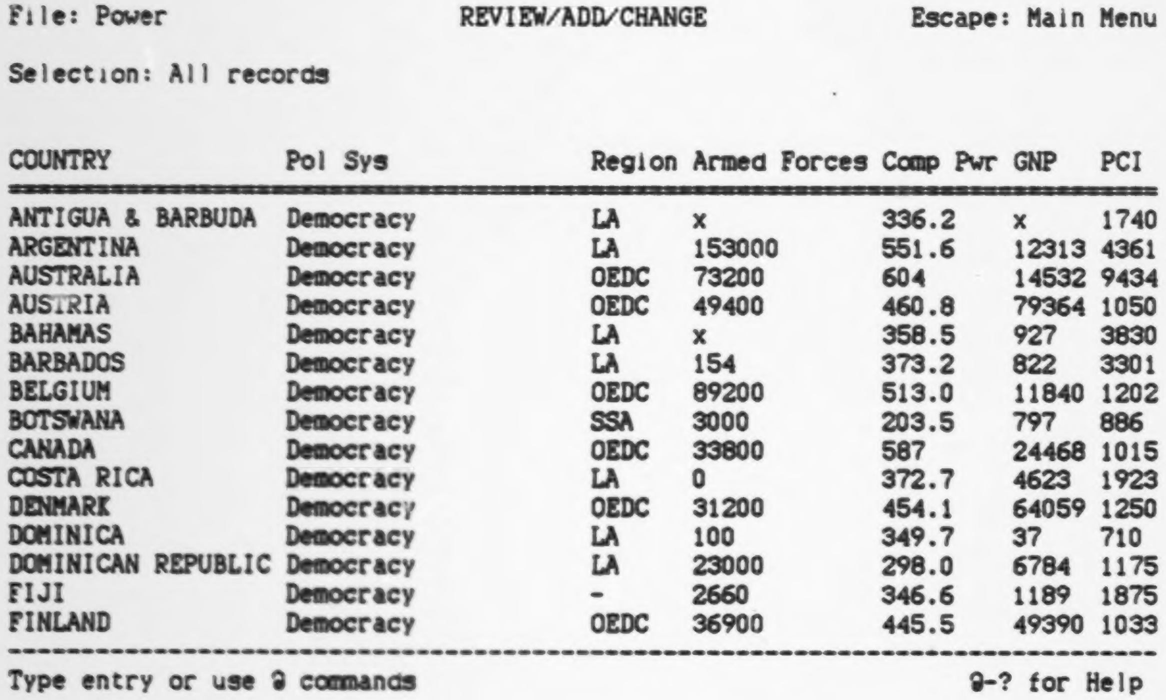
```
File: Power                    REVIEW/ADD/CHANGE              Escape: Main Menu

Selection: All records


Record 155 of 167
================================================================================
COUNTRY: UNITED STATES                   %Arable: 20
Region: OEDC                             Army: 790800
Sub-region: North America                Navy: 745000
Wld: 1      Pop/m: 232                   Airforce: 581000
Rank: 9                                  Sq km/1000: 9363
PCI: 11347                               Raw Pwr: 1944.0
Pol Sys: Democracy                       Comp Pwr: 3496.0
%Rural: 23                               TME/m$: 143974
%Cal Req: 138                            # Nukes: 18000/12000
%Sf Wtr: 99                              TVE/m$: 212300
%P Sch: 98                               TVI/m$: 243900
%S Sch: 97                               Energy P/MTCE: 2088
%Lit: 99                                 Energy C/MTCE: 2345
PQLI: 97                                 Armed Forces: 2116800
GNP: 2583700
--------------------------------------------------------------------------------
Type entry or use @ commands                                        @-? for Help
```



```
File: World Special            REVIEW/ADD/CHANGE              Escape: Main Menu

Selection: All records


Record 155 of 167
================================================================================
COUNTRY: UNITED STATES  Wld: 1    Independence: Britain        Indep Year: 1776
Rank: 9        PQLI: 97           Raw Pwr: 1944       Comp Pwr: 3496.0
GNP: 2583700  PCI: 11347          %Unemploy: 8.1             Ex Debt Pub: x
Paramilitary: x                        %pop < 15: 26        Life Expec: 74
Pol Sys: Democracy                     %Pop Growth: .9      %Inf Mort: 1.3
Resources: Oil Coal Fe Cu U Other minerals                        Count: 1
Exports: Machinery Chemicals Trans Equip Agri prod
Imports: Petro Machinery Trans equip
Trade Par Ex: Canada Japan Mexico UK FRG
Trade Par Imp: Canada  Japan Mexico UK FRG
Ethnic Groups: 87 European 11 Black .4 Amerindian .3 Japanese .2 Chinese
Languages: English Spanish
Religions: Christian Judaism

--------------------------------------------------------------------------------
Type entry or use @ commands                                        @-? for Help
```



```
File: Power                      REVIEW/ADD/CHANGE                  Escape: Main Menu

Selection: All records

COUNTRY             Pol Sys                Region Armed Forces Comp Pwr GNP   PCI
==================================================================================
UNITED STATES       Democracy              OEDC   2116800      3496.0   25837 1134
U.S.S.R.            Socialist              EESU   2750000      2242     12120 4564
JAPAN               Democracy              OEDC   245000       1421.0   10481 8975
WEST GERMANY        Democracy              OEDC   497300       1175.0   82488 1339
FRANCE              Democracy              OEDC   482600       1027.0   65412 1215
UNITED KINGDOM      Democracy              OEDC   327700       957.0    51600 9213
ITALY               Democracy              OEDC   370000       829.8    39392 7012
CHINA               Socialist              EASEA  1175000      960.9    30000 298
BRAZIL              Other                  LA     277100       865.5    24511 2002
CANADA              Democracy              OEDC   33800        587      24468 1015
SPAIN               Democracy              OEDC   317000       633.1    20804 5550
MEXICO              Other                  LA     120000       661.8    18161 2590
NETHERLANDS         Democracy              OEDC   102900       571.0    16122 1139
INDIA               Democracy              SASWA  1120000      497.3    15976 230
AUSTRALIA           Democracy              OEDC   73200        604      14532 9434
----------------------------------------------------------------------------------
Type entry or use @ commands                                          @-? for Help
```



```
File: Power                      REVIEW/ADD/CHANGE                  Escape: Main Menu

Selection: All records

COUNTRY             Pol Sys                Region Armed Forces Comp Pwr GNP   PCI
==================================================================================
ANTIGUA & BARBUDA   Democracy              LA     x            336.2    x     1740
ARGENTINA           Democracy              LA     153000       551.6    12313 4361
AUSTRALIA           Democracy              OEDC   73200        604      14532 9434
AUSTRIA             Democracy              OEDC   49400        460.8    79364 1050
BAHAMAS             Democracy              LA     x            358.5    927   3830
BARBADOS            Democracy              LA     154          373.2    822   3301
BELGIUM             Democracy              OEDC   89200        513.0    11840 1202
BOTSWANA            Democracy              SSA    3000         203.5    797   886
CANADA              Democracy              OEDC   33800        587      24468 1015
COSTA RICA          Democracy              LA     0            372.7    4623  1923
DENMARK             Democracy              OEDC   31200        454.1    64059 1250
DOMINICA            Democracy              LA     100          349.7    37    710
DOMINICAN REPUBLIC  Democracy              LA     23000        298.0    6784  1175
FIJI                Democracy              -      2660         346.6    1189  1875
FINLAND             Democracy              OEDC   36900        445.5    49390 1033
----------------------------------------------------------------------------------
Type entry or use @ commands                                          @-? for Help
```

GLOSSARY

%CAL REQ: Percentage of caloric requirements as defined by the World Health Organization.

COMP PWR: (composite power) Sum of raw power and points given for PQLI and PCI.

ENERGY C/MTCE: (consumption/millions of tons of coal equivalency) A measure of energy.

ENERGY P/MTCE: (production/millions of tons of coal equivalency) A measure of energy.

EX DEBT PUB: (external debt public) The amount of money owed by a state to other states or to national or international entities.

GNP: (gross national product) Total value of all goods and services produced by a state.

%INF MORT: (% infant mortality) Percentage of babies who die by age one.

# NUKES: Number of nuclear warheads a country possesses.

PCI: (per capita income) monetary amount found by dividing population by GDP(gross domestic product).

%POP <15: Percentage of total population which is under 15 years of age.

%POP GROWTH: Percentage of a population growth during a one year period.

POP/m: Population of a state expressed in millions.

PQLI: (physical quality of life index) Preferred by a number of observers as a more accurate measurement of a society's economic and social well-being as compared to PCI. Refers specifically to Literacy, Life expectency at birth, and Infant mortality rate. It is based on a scale of 0 to one hundred; one hundred being the best rating.

RANK: Refers to a composite socio-economic ranking based on the findings of Ruth Sivard; 1 being the highest rank.

RAW PWR: (raw power) Sum of points given for selected categories from these data bases.

%Rural: Percentage of population that does not live in cities of over 5,000 people.

%SF WTR: (safe water) Percentage of population who have access to potable water.

TME/m$: Total military expenditure in millions of dollars.

TRADE PAR EXP: (trading partners exports) States with whom exports are exchanged.

TRADE PAR IMP: (trading partners imports) States with whom imports are exchanged.

TVE/m$: Total value of exports in millions of dollars.

TVI/m$: Total value of imports in millions of dollars.

X: Negligible, not available, non-existent, or your guess is as good as ours.

THE CI^4 INSERVICE MODEL*

Judi Mathis, Jim McCauley & Seymour Hanfling

The Computer Innovations Integrated Into Instruction (CI^4) Project directed by Dr. David Moursund began in September 1985. The purpose of the three-year project is twofold: (1) to develop an inservice model for educating teachers in methods of integrating general computer software tools, such as databases, spreadsheets, graphics programs, science tool kits, etc., into their curriculum and (2) to develop a method for training inservice trainers in the use of the CI^4 model developed in step one. During the first year the project team has concentrated on developing the inservice model and materials in two areas: elementary schools and secondary school mathematics. The second year of the project will continue this teacher inservice development and extend it to include secondary school science. The third year will refine the materials from the second year, create materials for secondary school social studies and develop a training of trainers inservice model.

The ultimate goal of the CI^4 project is to bring about improvement in the classroom; this is a formidable task. The process of bringing about improvement through innovation in the classroom by staff development has been studied extensively (Dillon-Peterson, 1981; Fullan, 1982; Berman & McLaughlin, 1978). Even though there is no agreement on the "one" best way to do staff development, there is agreement on the need for the support and involvement of the major components of an educational system — the teachers in a school or department, the building administration, the central administration, the school board and the students' parents. The CI^4 project used this knowledge in formulating an inservice program.

First, we received the support of the central administration of the district within which the workshops took place. Second, we viewed the school (or department in the case of a high school) as the unit of change, not the entire district. We worked with groups of teachers from a building (or department). Finally, an administrator from each building was required to participate along with the teachers.

---

*The research described in this paper was supported by grant number TEI-8550588 from the National Science Foundation. The authors were graduate assistants employed by the grant during the 1985-86 school year.

As Wood, Thompson, and Russell (1981) point out:

> For staff development to have a lasting effect, the principal must be committed to the implementation of the inservice goals, participate in the inservice planning and activities, encourage other staff members to participate in training programs, and support and reinforce the implementation of new knowledge, skills, and strategies (p. 63).

During the past year the CI^4 project worked with two groups of educators. One consisted of the principals and a number of elementary school teachers from three schools. The second consisted of mathematics teachers from a middle school and a high school and an administrator from each building. There were about 17 participants (all volunteers) in each group.

Prior to the training, a needs assessment was conducted by interviewing all the participants. We acquired knowledge on a variety of topics, including the participants' current educational and personal computer usage; their access to computers, computer lab(s), and software; their views on educational uses of computers; and the areas in which they desired training.

Originally seven inservice sessions were scheduled. The sessions were conducted after school and usually at the computer lab sites of the the participating schools. (Two sessions were conducted on the University of Oregon campus.) The introductory and closing sessions were originally scheduled to be five hours long (with dinner provided) and the five other sessions were to be two hours in length, held every other week. The teachers found the first five-hour session to be too long after they had been teaching all day. We adjusted our schedule and shortened the last session to two and a half hours. In the future we are planning to conduct 16 hours of workshops in eight two-hour sessions.

The Inservice Model

> The wise leader knows that the true nature of events
> cannot be captured in words. So why pretend?
> Confusing jargon is one sure sign of a leader who does
> not know how things happen.
>
> The Tao of Leadership by John Heider

Inservice Design Principles

The participants in our inservices had a wide range of backgrounds and teaching environments. This lack of homogeneity is a common difficulty in computer-related inservice education, but it lead to the development of a significant premise of these workshops. These are not inservice trainings in the sense of teaching a specific teaching technique, computer management skill or even competence with a specific piece of software. These are educational workshops. They are meant to expand teacher knowledge and capabilities in the classroom. They provide an environment for exploring and learning about the applications of computers and software tools in the curriculum.

The following design principles are used to create this environment.

(1) Each session should offer the participants at least one idea that can be used in their classrooms. This idea might be an activity, a piece of software, a teaching style, or a management aid. The sessions are rich learning environments in which participants discover and experience many ideas and applications of computer-related activities and software tools. These environments provide many opportunities for participants to find ideas which are appropriate and relevant to their instructional situations. The process of discovery also helps build ownership and increases the likelihood of actual implementation in the classroom.

(2) Software should always be introduced in an instructional context. Rather than training participants in the details of a piece of software, the tool is introduced with classroom examples. Activities are designed so that the participants can be successful even if they have only partial knowledge of the software. This allows teachers to see that they need not understand all the nuances of a program before they use it in the classroom. The purpose of this approach is twofold. First, it helps teachers explore and gain an understanding of the instructional uses of a specific software tool. Second, it provides a model for teachers to experience an

activity and begin to adapt the activity to their own classroom.

(3) The participants should experience activities on two levels. First they are 'students' in a workshop — are the activities meeting objectives? What is being learned? What is being experienced? Is it enjoyable? Second is the analysis of the activities and the workshop itself — what activities have been chosen? Why have they been chosen and not others? How can they be adapted to the participants' classrooms?

Refection and discussion of these questions occurs at different times. The participants are briefly reminded to note and reflect on the first questions while they are doing an activity. These questions are then repeated during the closure discussion for an activity. The second set of questions is also discussed during closure for an activity and at closure for the entire session.

(4) The participants should work in groups. There are four reasons for this. First, Cox and Berger (1985) have shown that working in groups on the computer is more effective in solving problems than working individually. Second, the participants become accustomed to discussing computer-related curriculum matters with one another. This helps to build a school level and district wide resource base. Third, it allows the trainers to work with more of the participants. Finally, it allows the trainers to eavesdrop on participants' conversations and gain relevent information for conducting discussions and directing the remainder of the session.

(5) Each session should be structured to allow participants to discover methods and models of instruction. The participants demonstrate or gain an understanding of these during the debriefing of each actvity or at the session closure. This approach goes counter to prevailing inservice formats by avoiding an explicit statement of workshop objectives at the outset of each session. Our concern is to avoid creating a specific 'mind-set' in the participants that might deter them from making their own original observations or restrict them from making additional observations. This also allows them to experience the activities as their students might, thus providing valuable insights that can be gained in no other comparable manner.

This method requires the development of mutual trust. Through the activities the participants discover, learn and gain insights across many pedagogical domains. During the debriefings the trainers can aid the participants in understanding and adapting their insights. This discovery and debriefing-oriented model is stimulating, interesting, and successful.

(6) The sessions should be enjoyable!!!

Material Selection and Development of Activities

The focus of CI^4 project is to improve student problem solving by the integration of software tools (graphics, data bases, spreadsheets, science kits, etc.) into the curriculum. Thus the theme of problem solving is interwoven throughout the design of the inservices. Workshop participants can explore and experience the use of these tools by solving problems presented in the inservices. These experiences also encourage the participants to discover new ways to pose problems and develop new types of problems to be solved.

Due to the wide range of backgrounds and experiences of the participants, as well as gender differences, careful selection of materials and activities is important. However, selection of software is also limited by practical constraints. Our selections are based upon availability, quality, utility in the particular workshop situation, appropriateness to the grade and concept level, and effective cost. The last item refers to software that is under district license, in the public domain or allows multiple loading. (Some software companies give special permission for multiple loading to educators performing workshops.)

After the design of activities and selection of software, performance aids and learning aids (worksheets) are developed. Performance aids contain the basic information necessary to use a piece of software (keystroke commands, data retrieval and printer commands, for example). Many participants find that they can use unfamiliar software without much instruction if they are supplied with an appropriate performance aid. Participants can also learn to use some software through on-line tutorials. Learning aids contain problems which increase in complexity and software knowledge. They therefore may contain software comments where appropriate (e.g. how to print a graph). These worksheets span the range from very specific instructional sequences to open-ended explorations.

Along with the performance and learning aids numerous sample lesson plans are provided. These resources are important in assisting the participants in transferring workshop concepts to their classroom. Lesson plans can be adapted, extended and used as models by the participants. This reduces the amount of effort required to develop and transfer new activities to the classroom.

Organization and Development of an Inservice Script

In organizing the sessions we tried various workshop methods and activity sequences. The ones we found successful were those that embodied our assumption that instruction is a process that by its nature is dynamic. Decisions are made continually during the training sessions through the interactions of the trainers, the participants and the content being presented. Thus the framework of these sessions must be flexible. The same session done with different groups of participants might begin in the same way, but then, based upon their responses and needs, proceed along different paths. Many times the participants are offered a choice of options or allowed to offer their own suggestions on the next step in the inservice. This dynamic process is difficult to capture in words. In the following discussion it is important for the reader to keep in mind that these workshops are a process that are designed to be flexible.

Sequencing of Activities. The sequencing of all activities enhances concept attainment and assists in the transfer of these concepts to the classroom. The sequencing of activities begins with the most concrete activities and progresses to the more abstract.

The typical sequence for an activity is:

(1) Explore. Allow participants time to "play" so that they may develop an intuitive understanding of the software or activity.
(2) Experience. Work on the software or activities in an instructional format that models classroom presentations.
(3) Discuss. Debrief the activity, with particular attention to the participants' feelings, experiences, attitudes and ideas for instructional applications.
(4) Closure. Relate this activity to the objectives of the session or previous sessions. During these closures integrate previous participant comments as frequently as possible.

Sequencing within a Session. We begin each session with a hands-on activity, either computer-based or a problem solving activity, that is approximately 5 -15 minutes long. (Sessions were usually conducted in a lab that allowed for no more than three people per machine.) This instant involvement is very useful in setting the tone for the day's session. The debriefing of this opening activity provides an opportunity to discuss the focus and goals of the session.

The next 30-40 minutes is spent in exploratory activities on the computer with the participants working in groups, generally in pairs. This might be directed by the trainer or through performance and learning aids. The focus is on the specific objectives of the activity. Participants can become involved in trying to "solve the puzzle" or "beat the computer" and it is sometimes necessary for the trainers to draw the participants' attention to the instructional aspects of the activities. The participants may finish an activity at a later time and at their own pace.

The debriefing of all activities is extremely important. As described above, explanation and debriefing of all activities follows rather than precedes the activities. This allows the participants to experience an activity in a situation without a trainer-induced 'mindset'and places them in a position similar to what their students might experience. Encouraging the participants to generate ideas contributes to the transfer of ideas from the workshop to the participants' teaching situations. It is more likely that teachers will try new classroom practices if they have experienced these new practices in the workshops and then analyzed and reflected on them.

The remainder of the session is spent intermixing off-line and computer-based activities. The participants enjoy the integration of non-computer activities into the sessions. The concrete nature of these activities provides a bridge to the abstract nature of the computer.

Final closure for an entire session provides a transition from the workshop setting to the classroom. Our experience indicates that the integration of software tool activities into classes seems to take place if the integration (1) allows students to learn better, faster or in more exciting ways, (2) allows students to work with more important concepts than they are now learning and (3) is not appreciably more difficult for the teacher than present instructional practice. Discussions during the final closure are useful in generating ideas that illustrate these points.

Once the above ideas have been used in planning a session or series of sessions, it is important to review the entire set of activities. Are the transitions smooth? Are the activities building upon each other? Are the original objectives still being met? The last question can sometimes be overlooked as "exciting" activities and software hide or even change the original objectives. Finally, is this still a workshop that is interesting and enjoyable to lead and experience?

Workshop Techniques. Like any teachers, we use a variety of techniques during the sessions. The following are those we find useful and consider the most important.

(1) Model classroom activities and appropriate teacher behaviors. We generally teach the way we have been taught. The importance of modeling appropriate teacher behaviors cannot be overstated in helping the participants to integrate

new ideas into their classes. Just discussing how activities can be done in the classroom is insufficient; they must be experienced to be understood.

(2) Identify and make use of participants with knowledge of the workshop contents. Forming groups where at least one of the participants has experience with a piece of software can help make workshops go smoothly. Establish peer support by publicly validating participants' knowledge and encouraging peers to consult each other for answers. Remind participants that they are experienced educators who bring a multitude of skills to the inservices. (An analagous situation for teachers is using students in their class who are experienced with software to act as helpers or teachers to other students.)

(3) Obtain feedback from the participants at each step of the workshop. Be responsive to their needs and when appropriate modify the workshop plan. When necessary, help the participants reevaluate their expectations in order to gain the most from the workshop. For example, when a wide range of teacher backgrounds is present at a workshop, a specific piece of software may not fit a teacher's class. Shift the focus of the teachers' awareness to examining the software for its strengths and weaknesses — is it easy to use? Is the feedback appropriate? Does it allow for exploration?

(4) Occasionally interrupt participants in the middle of an activity. Research evidence suggests that this can actually contribute significantly to learning. This interruption provides a time for the participants to note and reflect upon their experiences during an activity. Later discussions encourage the participants to analyze the activity from two perspectives — that of a student (their experiences while using the software) and that of a teacher.

(5) Use open-ended questions that encourage teachers to reflect on the session's activities. Convergent questions can cover the content and analysis of the session's activities, pacing and sequencing. Divergent discussions can cover transfer of the activities to the classroom and the possible ramifications on the curriculum and individual classes. These types of questions are important and cannot be rushed through. Time for reflection and analysis helps the participants gain the ownership that is necessary for integrating new ideas into their own educational setting.

(6) Finally, when the trainer does not have an answer to a question the best response is "I don't know." However, there may be participants that can provide information, just as the teachers may have students in their class who can help them with a piece of software or activity.

Reference Materials, Handouts and Log Sheets

Each participant receives a set of materials consisting of relevant journal articles, software reference lists, classroom ideas and public domain software at the first session. Participants are referred to elements of this collection whenever related activities or discussions take place.

The participants are asked to keep a voluntary log of their computer-related activties including: classroom lessons, work with individual students, personal use, readings and discussions with colleagues. This log form can also be used to ask the trainers for information or help. The trainers respond to questions, suggest software the participants might wish to preview and return the logs. These forms are an important feedback mechanism. It is an efficient way for the participants to communicate with the trainers and for the participants to have a record of their computer-related activities.

School Visits

Based on the project's experience, classroom visits will be included as part of the experience of the inservice program. This can be done in a variety of ways. A trainer may model a lesson in a participant's classroom. Both the participant and trainer may teach a class together. The trainer may observe the participant teaching a lesson. Many teachers feel that this type of interaction is helpful.

Closing Comments

We have spent many hundreds of hours working on the CI^4 project. It has been difficult to summarize what we have learned. The main reason was stated earlier — education is a process that is dynamic. The only way to learn a process is to do it. What we have presented here is a framework that has been effective. However, this is not the end of the development of an effective inservice model. Both formative and summative evaluations have taken place or are in progress; as the results become clear, the model changes. Over the next two years we expect to learn more about the process of integrating computers into the curriculum and use that knowledge in refining the inservice design. One of the joys and frustrations of education is that it is constantly evolving. So is the CI^4 inservice model.

Bibliography

Berman, P. and McLaughlin, M. Federal Pograms Supporting Educational Change, Vol. VIII, Implementing and Sustaining Innovations. Santa Monica, California: Rand Corporation, 1978.

Cox, D. A. and Berger, C. F. "The Importance of Group Size in the Use of Problem-Solving Skills on a Microcomputer." Journal of Educational Computing Research, 1985, 4, pp. 459-468.

Dillon-Peterson, B., ed. Staff Development/Organization Development. ASCD Yearbook, 1981.

Fullan, M. The Meaning of Educational Change. New York: Teachers College Press, 1982.

Heider, J. The Tao of Leadership. Atlanta: Humanics New Age, 1985.

Wood, F.H., Thompson, S.R., and Russell, F. "Designing Effective Staff Development Programs." In Staff Development/Organization Development (B. Dillon-Peterson, ed.). ASCD Yearbook, 1981.

THE APPLE CLASSROOM OF TOMORROW

Dr. Samuel K. Miller

Introduction

Imagine trying to demonstrate the benefits of pencils or books if students only used them several times a week. Computer-using educators face a similar dilemma. Based on a sample of 2,300 schools, findings from the "Second National Survey of Instructional Uses of School Computers" conducted by Henry Jay Becker of Johns Hopkins University, indicate that one million computers are now in elementary and secondary schools (*Electronic Learning*, 1986). Becker's study suggests that a quarter of American schools now have enough computers to teach between a half and one full class of students at a time. These figures represent a major increase in the number of computers in the schools compared to several years ago; however, the biggest problem reported by teachers continues to be the lack of available equipment.

In spite of limited resources, many educators continue to be enthusiastic about using computers. Skillful and persistent teachers nationwide have produced many fine examples of the computer's educational potential under less than optimal conditions. But consider for a moment what teachers and students might accomplish if they had sufficient computer resources, a classroom where computers and other instructional technologies were as accessible as pencils and books. The Apple Classroom of Tomorrow at Colin Kelly Middle School in Eugene, Oregon, is one such classroom.

Project Background

The Apple Classroom of Tomorrow is a research and development project sponsored by Apple Computer, Inc. Conceived and directed by Dr. Martin Engel of Apple's Advanced Development Group, the project's goal is to create instructional environments in which every student has complete access to a classroom and home computer in order to learn how to maximize computer use for the entire school curriculum. To accomplish this goal, Apple has provided hardware, software, and personnel support during the 1985-86 and 1986-87 school years to a select group of seven elementary, middle, and senior high school classrooms in school districts throughout the United States. The project is a partnership program--not a grant--and each district is expected to contribute funding and resources to the project.

The Classrooms of Tomorrow are set in school districts of computer-using leaders in public education, but each site is different with regard to grade level, socio-economic factors, and instructional focus. Although the school districts participating in the project have signed an agreement with Apple, each project site has autonomy in organizing curriculum and instruction and in staff selection. At present the Apple Classrooms are not considered a promotional vehicle for Apple products; however, the company plans to field-test new products in selected sites and will conduct a formal project evaluation.

Eugene Project Overview

Students and Staff

The Eugene School District has 44 schools that serve over 17,000 students. During the 1985-86 school year, Eugene's Apple Classroom of Tomorrow project was conducted at Colin Kelly Middle School. Colin Kelly provides instruction for 425 students in grades six through eight. The school population is primarily white and consists of a large number of students from low socioeconomic families.

Eugene's Apple Classroom of Tomorrow consisted of 13 boys and 13 girls randomly selected from the population of sixth-graders enrolled at Colin Kelly Middle School. Throughout the project, parents had the option to place their child in the regular middle school program. During the school year a girl in the project moved and a boy was placed in the regular program. The class's academic performance level was heterogeneous. In tests administered during September 1985, students in the classroom scored from the 8th to 95th percentile on the mathematics and reading sections of the California Test of Basic Skills (CTBS).

Michael Caley and I shared instructional responsibility in the Apple classroom; however, we did not teach the class at the same time. Both of us are experienced elementary and middle school teachers with extensive backgrounds in curriculum development and computer education. We have previously worked together and share a common instructional philosophy.

Resources

During the 1985-86 school year, Eugene's Apple Classroom of Tomorrow provided 26 sixth-grade students at Colin Kelly Middle School with 26 Apple IIe systems for use during the school day and an Apple IIc microcomputer for use at home. Eugene's project goals also extended beyond using computer technology to include the integration of videotapes, videodiscs, and calculators in addition to traditional instructional materials.

Eugene's Apple Classroom of Tomorrow is also the recipient of a $35,000 networking hardware and software award from Corvus Systems, Inc. Major software contributors to the project included Scott,Foresman and Company, Mindscape, Scholastic, Sunburst, DLM Teaching Resources, Holt, Rinehart and Winston, Sensible Software, and Science Research Associates.

Curriculum and Instruction

Students received instruction in the Apple Classroom of Tomorrow during a five-period block. The main subjects taught during the block were mathematics, science, language arts, social studies and reading. Students also received physical education and an elective class during the two periods they were not in the Apple classroom.

When the project's computer equipment arrived in November, the students began daily keyboarding instruction for eight weeks. By June the class average on a timed touch-typing test was 24 words a minute with two errors or less. With students' individual keyboarding rates ranging from 14 to 47 words a minute, all of the students were facile at using a word processor for their written work.

The curriculum resembled conventional sixth-grade subject matter with the primary difference being the use of technology to supplement or supplant the traditional instruction. For example, students learned, practiced, and applied the school district's goals and objectives for sixth-grade geometry via the *Apple Logo II* programming language. Another example is composition instruction; all of the students' written work was composed and checked using the *AppleWorks* word processor and *Sensible Software*'s spelling checker.

Computers were rarely used for initial instructional purposes. Both teachers preferred to preface student work with teacher-directed instruction followed by closely supervised work at the computer. However, an outstanding videodisc program, "Mastering Fractions" was used for initial instruction in basic fractions operations.

As the school year progressed, the amount of class work centered around computers increased. The shift from conventional instructional materials to computer-based instruction can be attributed to an increase in expertise and confidence by the project students and teachers. This gradual evolution in using technology in the classroom emphasizes a frequently recognized variable in any instructional innovation--the willingness to patiently wait for change to occur.

A visitor to the classroom at the end of the school year would have observed a group of students who were comfortable using computers to create graphs, draw pictures, write compositions, play games, or devise data bases. The visitor would also notice that the students appeared to enjoy their work but still behaved like "regular" sixth-grade students. Not surpisingly, Eugene's Apple Classroom of Tomorrow had its share of students who wouldn't listen to directions, as well as students who would forget to complete their homework. The students felt pleased about the opportunity to work with computers but did not spend all of their waking hours enamored by their good fortune. They simply regarded computers as another appliance.

The Home Connection

Before computers were sent home with students, their parents were required to attend two training sessions. The project staff taught all of the parents how to assemble and use their home computer. If parents already had or could afford to insure their child's computer, they did; otherwise the computers were uninsured. This posed no problems and all of the machines were returned intact at the end of the school year.

Additional parent training classes were offerred but were cancelled due to lack of interest. Several parents assisted in the classroom or served on the project advisory committee. All of the parents were enthusiastic about the project and were pleased with the experience their child received.

Students used their home computers to practice keyboarding using *MicroType: The Wonderful World of Paws* or to complete homework assignments using *AppleWorks* or *Apple Logo II*. Some software was loaned to students during the year for independent use. If students experienced problems at home using a program or their computer, they would call a project staff member for assistance. Students carried their data disks between school and home in a plastic case and printed their work at school. This procedure worked quite well except for some students who consistently forgot their disks. Some things never change!

Project Evaluation

The Eugene School District's research department assisted the project staff in devising and implementing an evaluation plan. Eventually the research department will issue a report summarizing the project's first year. The project evaluation included a comparison of the Apple Classroom of Tomorrow students with sixth-grade students in the regular Colin Kelly Middle School program. Several measures were obtained, including standardized tests, compositions, attitude surveys, attendance, and behavioral records. Additional data were collected by an ethnographer assigned to observe and document life in the Apple classroom.

A preliminary review of the data indicates that students in the Apple Classroom of Tomorrow performed as well on the standardized tests as their peers who received instruction in the non-computer classroom. The data from the writing assessment indicate that the Apple classroom students scored higher than the comparison group in writing quantity and quality. In general there appears to be no significant difference in student attitudes or attendance data.

At this point the very least that can be said about the data is that the students in the Apple Classroom of Tomorrow had as productive an instructional experience as students in the comparison group. There are additional factors that indicate the success of Eugene's Apple Classroom of Tomorrow. For example, both project staff members observed that the students appeared to work exceptionally well together as a group. This can probably be attributed to the numerous opportunties they had during the year to use computers to work on cooperative projects. It is important to note that significant instructional time was dedicated to teaching students critical prerequisite skills such as keyboarding, word processing, and Logo. The benefits of learning these skills will only be realized when students consolidate and apply what they have learned. This is an important goal for the second year of Eugene's Apple Classroom of Tomorrow project.

Future Plans

With assistance from Apple Computer, Inc., Eugene's Apple Classroom of Tomorrow project will continue with the addition of a second classroom during the 1986-87 school year. A new group of 26 sixth-graders will comprise one classroom, and the second classroom will consist of 21 "graduates" from the 1985-86 Apple classroom. Patsy Sparks, a teacher at Colin Kelly Middle School, will be the third project staff member.

The opportunity to obtain longitudinal data is imperative if we are to know what ramifications exist for students who advance beyond one year in a technology-rich classroom. Two classrooms will also permit some instructional groupings between the sixth and seventh grades based on student needs and performance levels.

Several new goals will also be added to the project. They include increasing the amount of instruction that is computer-mediated, providing students with opportunties to complete student-centered projects, and developing curriculum materials for dissemination that use *Apple Logo II, AppleWorks,* and *MousePaint.*

Summary

As a teacher and project coordinator in Eugene's Apple Classroom of Tomorrow, I can report without hesitation that my experience with this project has been extremely thought provoking and time consuming. In the future I plan to write a more personalized account about living and learning in a technological fishbowl. I recognize that this report is compressed and is a somewhat prosaic view of a project with important implications for anyone intrigued by the prospect of improving education via technology. Furthermore, it ignores the human factors that make any endeavor a success or failure and is no substitute for living in the classroom of tomorrow--today. Visitors to the project are welcome and should contact me after October 15, 1986, at the address listed below.

Dr. Samuel K. Miller, Project Coordinator
Apple Classroom of Tomorrow
Colin Kelly Middle School
850 Howard Avenue
Eugene, Oregon 97404
Phone: (503) 687-3224

References

"Survey Maps Computer Use: University Study Tracks the Role of Micros in the Classroom." *Electronic Learning.* 5, 4 (January 1986), pp. 12-13.

THE FUTURE OF COMPUTERS IN INSTRUCTION

David Moursund
ICCE, University of Oregon
1787 Agate Street
Eugene, Oregon 97403

Audience and Purpose

This paper about the future of computers in instruction is intended for a number of different groups of people, with some people belonging to more than one group.

1. Students, or people planning to return to school; predictions about the future may influence their choice of coursework or major.
2. Teachers; predictions about the future may help them in making changes to the content of their courses and the methods of instruction used in these courses.
3. Educational leaders and planners; predictions about the future may assist them in long term planning.
4. People involved in educationally-oriented businesses; predictions about the future may help them in making wise business decisions.
5. The intellectually curious; it is fun to speculate about the future and to listen to other's speculations.

This paper is divided into a number of sections. First there is a general discussion of the basis for my predictions. The astute reader will notice that this paper is **not** research-based (notice the lack of a bibliography at the end of the paper). However, there is some underlying logic and evidence to support the predictions. Most of the predictions are statements about current trends and are not quantitative in nature.

Next the paper discusses some trends in hardware, software, networking, computer science and overall applications of computers in instruction.

The paper concludes with several recommendations. These recommendations are based on the overall prediction that computers will lead to profound changes in our educational system.

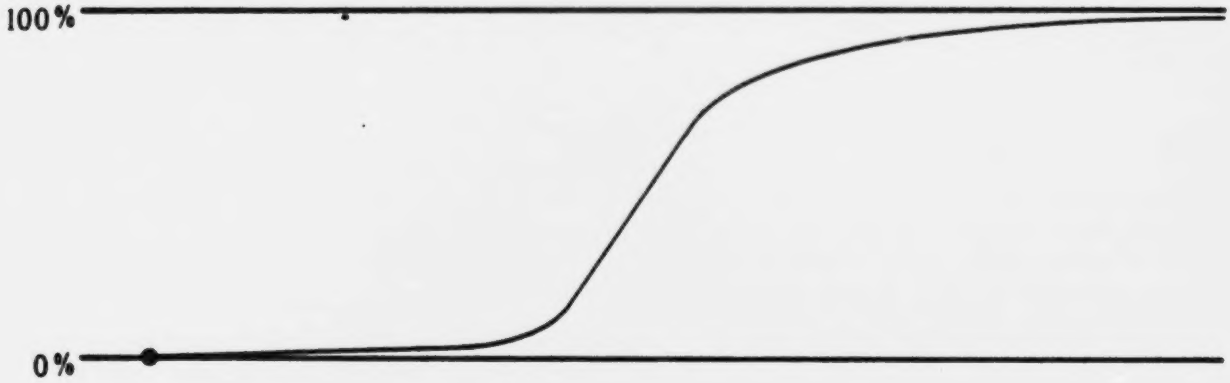
S-Shaped Growth Curve

I had already graduated from college when I had my first hands-on experience with a computer in 1959. The first computer I wrote programs for was a first generation (vacuum tube) machine called the IBM 650. If two ten-digit integers had been placed in the right registers, it could add them in a thousandth of a second! When suitably programmed to calculate a square root of a ten-digit integer, it could extract the square root in an eighth of a second. Needless to say, students studying this machine were quite impressed by its "blinding speed." This machine served the entire research and instructional needs of the University of Wisconsin, Madison. Now, of course, millions of people own personal microcomputers with far more compute power.

I have been a computer educator for more than twenty years. During that time I have taught a wide variety of courses and students, visited with educators throughout North America, and studied and read extensively. Thus, I have been deeply involved as computers have come into our schools. Many of my predictions are based on common sense extrapolations of what I have observed over the years.

Computers are an example of technology, and many other technologies have come into our society. For example, the telephone was invented by Alexander Graham Bell in 1876. After its invention, a technologically-based product then undergoes the work needed to produce a marketable product and then is brought to market.

Under free market conditions, product acceptance and distribution is characterized by an "S-shaped" growth curve.



Initial growth in use and acceptance of a new product is slow because the product is not know to many people, the product is in short supply and it may be quite expensive, the necessary supportive infrastructure is not in place, etc. For example, when television was first invented there were no television stations, sales outlets or repair places.

Notice the period of rapid growth in the middle of the S-shaped growth curve. Both the number of years of rapid growth and the rate of rapid growth vary with the product. But eventually the market becomes saturated.

The instructional use of computers seems to be following an S-shaped growth curve. In precollege education we seem to be just beginning the period of most rapid growth. Some authors suggest that the number of computers in precollege education is now doubling every 15 months.

However, the reader should be aware that computers in schools may not continue to follow the S-shaped growth curve model. The education market is not like the general consumer-based market. Funding for education is very tight, so computers-in-education may not follow a standard free market growth pattern.

Four Eras

There have been a number of writers who have examined and characterized major trends in human societies. They have identified four major eras: hunter-gatherer, agricultural, industrial and information. Until about 10,000 years ago all societies were hunter-gatherer in nature. Then the agriculture era emerged in various parts of the world. About two hundred years ago the agricultural era began to give way to the industrial era in some parts of the world . Finally, the information era began to emerge about 30 years ago in some industrialized societies.

The label "Information Era" is misleading. What has happened in the United States during the past 30 years has been a massive shift from a manufacturing economy into a service economy. For example, the number of fast food stores has grown very rapidly, as has the number of clerks working in these stores. These are certainly not high-tech, information-oriented jobs.

But information-related technology is certainly an important part of our information era. Who can help but be aware of the rapid growth of IBM, whose sales last year were fifty billion dollars? And computer-related technology has transformed many industries. For example, our telephone system is much better than it was 30 years ago, due to computerized switching and other computer equipment, satellites, fiber optics and so on.

Computers and computer-related technology are playing two ever increasing roles in our information era. First, in many situations computers are a productivity aid. These situations range from a writer using a word processor to a librarian searching computerized data banks to a computerized robot helping to assemble automobiles. Second, computers are being built into many devices such as automobiles, microwave ovens and television sets. Sometimes computer technology makes possible entirely new devices, such as electronic digital wrist watches and hand held electronic calculators. These two general categories of computer use--**productivity** and

products-- are now well established and will surely continue. They can serve as a solid foundation for predictions about the future of computers in education.

Computer Hardware Trends

To a large extent the computer field has been driven by hardware progress, and this progress is showing no signs of slowing. We will examine the trends here and make a couple of general predictions. You should notice that these predictions are made quite independently of whether computers become of increased importance in education. That is, the education market is not yet large enough to be a major driving force in the development of hardware.

The ENIAC, which became operational late in 1945, was the first general purpose electronic digital computer produced in the United States. It contained 18,000 vacuum tubes, and these vacuum tubes were at the very heart of the computational circuitry.

The transistor was invented in 1947, and one can think of a transistor as being roughly equivalent to a vacuum tube. It took about ten years before transistorized computers began to become available. At that time transistors cost about the same as vacuum tubes, but used less power and were more reliable.

In much less than another ten years the integrated circuit was in mass production. The IBM 360 series of computers that came out in the mid 1960s made use of integrated circuitry and were called third generation machines. A historical note seems appropriate. When I arrived at the University of Oregon in the fall of 1967 they had just installed an IBM 360 model 50. The UO was quite proud of this new machine with its high speed, 256K-byte memory, multiple tape drives, and two five-megabyte disk drives. Later, at a cost of about a quarter million dollars, eight 27-megabyte disk drives were added to the system. The total system cost about a million dollars.

In terms of CPU capability, primary storage and secondary storage, the cost of computers has come down by a factor or perhaps 500 during the past 20 years. The personal microcomputer I now own has more primary storage and raw compute power than the IBM 360 model 50. There has also been substantial progress in software, a fact discussed later in this paper.

The trend of packing more and more circuitry into a small chip has continued unabated. Chips containing the equivalent of more than two million transistors (these are 128K-byte memory chips) are now in mass production. Researchers seem confident that a further gain by a factor of 32 or 64 will be achieved by the year 2000. The next step seems clear. Researchers in a Texas Instruments lab produced a 512K-byte memory chip in summer 1985. Such a product should be in mass production by a number of companies before 1990.

The hardware progress has let to single chip 32-bit CPUs in 1981, second generation 32-bit CPUs in 1985, and a variety of still faster 32-bit CPUs during the past year. Considerable progress in commercially available parallel processors, involving the interconnection of multiple CPUs and primary memory, has occurred in the past year. The "neural-network" type of computer is just now starting to become commercially available. In this type of hardware design a large number of processing units are interconnected, much like neurons are interconnected in a brain. Certain types of problems can be solved very rapidly by such hardware.

Computer hardware has made so much progress that it is now possible to design special circuitry and build special-purpose computers specifically for a particular application. At Carnegie-Mellon University, for example, a special-purpose chess playing computer has been designed and built. This machine employs 64 special-purpose processors (one for each square on a chess board) in addition to a general purpose processor which runs the master program and coordinates the overall efforts. The system can examine 175,000 possible chess moves per second. It recently won the North American computer chess title.

The trend toward greater packing density in computer circuitry has made portable microcomputers possible. Portable microcomputers are now readily available, ranging in capability from not much more than a sophisticated pocket calculator up to machines fully as powerful as medium priced microcomputer systems. Display screens and batteries remain major problems. Slow progress is occurring with display screens. Circuitry using less power is being developed, and batteries are very slowly improving. It seems clear that the cost of portable microcomputers will slowly decline, as their compute power gradually increases. The market for medium capability portable microcomputers is still modest in size, although there have been several very large sales this year. Thus, prices are declining only slowly.

To summarize, we can predict with considerable confidence that computer hardware--CPU and primary memory-- will become more and more powerful and that the real costs of a given amount of compute power will continue to decline quite rapidly.

Besides raw compute power, rapid progress continues in secondary storage devices and in input/output devices. We give a brief discussion of each of these areas.

Secondary storage is making very rapid progress in both magnetic storage devices and optical storage devices. Floppy disk technology continues to progress. Kodak recently announced plans to begin marketing a 10 megabyte floppy disk system this year. Hard disk systems are rapidly decreasing in price and increasing in capacity. The cost of a 20 or 30 megabyte hard disk system for a microcomputer is under a thousand dollars. Much larger capacity hard disk systems are now quite common on microcomputers.

Optical storage technology is now a reality, but still in its infancy. The 550 megabyte laser disc system (the disc itself is only 4.72 inches in diameter) is already finding many commercial applications. For example, some poison control centers are using laser disc systems, and they find they can do better searches (using boolean logic) and can access information much faster from such a system then they can from microfiche.

Write once, read many time (WORM) technology is now on the market. One application for such technology is for preparation of large data banks that will be distributed (for example, by mail) to a number of sites. Another application is for archival storage. The write/erase technology for optical disks has been demonstrated and is headed toward the commercial marketplace. All of this optical storage technology will help provide increased access to larger and larger data banks.

In the input/output arena, there are two driving forces. One is to improve the human-machine interface for computer users. This can be done, for example, by suitable use of voice or sound output, voice input, graphic displays, mouse, glove, touch screen, etc. The second is to reduce the cost and improve the quality of "conventional" input devices and hardcopy output display devices.

Voice input has represented a considerable challenge to researchers in computer science. Significant progress has occurred, but the problem has proven to be a major challenge to researchers in artificial intelligence. It is now clear that eventually there will be systems that can handle large vocabulary, connected speech input with considerable accuracy. While business applications are clear, imagine the applications to teaching reading and writing to young children!

The past few years have seen a strong trend toward high resolution graphics displays (often with color) for microcomputer systems. Humans have considerable skill at recognizing visual patterns and in processing visually-presented information. It seems evident that high resolution graphics will play an ever increasing role in the computer field.

Computer hardcopy output devices (impact printers and laser printers) continue to make rapid progress. Just 10 years ago a good quality 10 character per second upper case-only computer printer cost about $3,500. Now one can purchase impact printers that are more than ten times as fast, much produce much better images, handle graphics--and cost about a tenth as much when the price is adjusted for inflation. However, such rapid progress in impact printer technology will not continue.

Laser printers are now coming to the fore. The past couple of years have seen rapid decreases in price and a number of new products. A period of rapid progress is occurring, and it should continue for quite a few years.

Software Trends

Software progress over the past 40 years has been steady, but certainly not as spectacular as the hardware progress. We will briefly discuss three aspects of software:

- Programming languages and operating systems
- Applications software
- Programming systems

Rapid progress in programming languages occurred quite early. From the early use of machine language it was a relatively quick step to assembly language. Higher level languages such as Fortran (developed 1953-1957) represented major progress. Many other high level programming languages were developed to fit the needs of specific groups of users. One interesting aspect of this is how long a language lasts once it gains a significant base of users. For example, Fortran is still alive and well, although current versions are a far cry from the early versions. I noticed recently that a Fortran compiler is now available for the Macintosh! BASIC seems to have equal persistence, with new versions recently appearing.

Since the late 1960s there has been a strong trend toward structured programming languages such as Pascal and C. The productivity of an individual programmer is not greatly enhanced by these languages as compared with programming in Fortran. But teams of programmers can much more easily work together, and the resulting products are more maintainable. In terms of maintainability and modifiability, structured programming languages have led to progress by a factor of perhaps three to five.

The quest for still better programming languages continues. One result is Ada, which has received strong support from the US military system. However, many professional programmers suggest that it represents a great leap backwards. My guess is that we will not make very rapid progress in developing still better programming languages.

The initial progress in developing better operating systems parallelled progress in developing better programming languages. We now take such things as a time shared operating system and mouse-based manipulation of files on a microcomputer for granted. In some ways we have witnessed a merging of programming languages and operating systems such as programming in C in a Unix operating system environment.

The programming language and operating system area faces a major challenge of parallel processing and neural-network systems. The goal is to have programming languages and operating systems aids to help the programmer effectively use these types of hardware systems. These appear to be formidable problems in computer science, and progress is likely to be slow.

Next we look at applications software. Here we sometimes forget the progress that had already occurred 20 years ago. I made use of a quite sophisticated linear programming package to compute some examples in my PhD dissertation during 1962-63. My wife made substantial use of statistical packages to do her PhD research at the same time. Now, of course, such statistical packages run on microcomputers and are a little more user friendly.

There seem to be two clear ideas in the computer applications area. First, any application running on a mainframe computer will eventually be available on a microcomputer. The user friendliness will be improved to increase the mass marketability of the product. Here it is worth mentioning IBM's recent announcement that they have produced an IBM 360/370 CPU chip. This suggests that eventually much of the IBM software library will become available to microcomputer users.

Second, artificial intelligence is now producing a large number of aids to problem solving. This artificially intelligence applications software is increasing the range of problems that can be handled by computer. Many hundreds of such products are coming to market. This trend will surely continue through the work of artificial intelligence researchers and as the needed compute power becomes more cheaply available.

Voice input provides a good example. This problem has challenged computer scientists for a good many years, and still is far from a satisfactory solution. But enough progress has occurred to allow the production of commercially viable products. Such products are now coming to market, and we can expect significant progress over the next decade. Ten years from now we will be seeing many young children using voice I/O as they learn to read and write.

Networking

As mentioned earlier, one characterization of the Information Age is more and better access to information. The worldwide telecommunications system is growing very rapidly, aided by satellites, microwave and fiber optics. Local area networks are now common and this is an area of very rapid growth.

It is hard to appreciate the potential here. Eventually the copper wires connecting our home telephones will be replaced by fiber optics. The capacity of a pair of fiber optics, each as thin as a human hair, is thousands of times the capacity of a pair of copper wires. For example, fiber optics can support high quality interactive color television.

Rapid progress is occurring in fiber optics. Recent research results suggest the possibility of a single fiber optic having the potential to carry ten million simultaneous two-way telephone conversations. This capacity would allow a full length movie to be transmitted in a fraction of a second.

A combination of improved networking and improved large scale database storage devices will have a profound impact on libraries and the publishing industry. We must realize that libraries and the publishing industry are very large, relatively slow moving institutions. Thus, change will be slow but steady. Already we can see significant changes in research libraries and in the storage/retrieval of literature to support research.

Computer Science

The discipline of computer science is now well established. There are hundreds of computer science departments at the college and university level. There are national recommendations on curriculum and there are textbooks to support these recommendations. A very large number of college students are majoring in computer science. Interesting, however, enrollment has peaked and is now declining. Perhaps students are seeing that computer science is both a hard major and no guarantee of a high paying job.

During the past thirty years we have seen computer science course content rapidly filter down from the graduate level to the undergraduate level and even into the precollege level. List processing and recursion in Logo, a language designed for elementary school students, certainly illustrate this point. The high school Advanced Placement course in computer science contains many topics that used to be studied at the advanced undergraduate level.

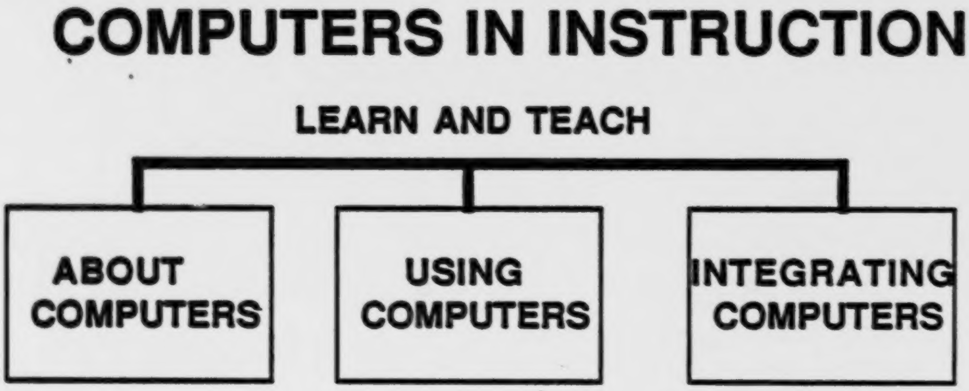
While the discipline of computer science is beginning to mature, it seems evident that rapid change will continue. An excellent example is provided by the Prolog programming language. A few years ago it was primarily of interest to graduate students specializing in artificial intelligence. Now it may be encountered in a freshman course for computer science majors. It is even being used with grade school students in a research project in England.

College and university computer science departments face an interesting challenge. As computer science matures, it becomes a more rigorous, mathematically-oriented subject. It is a discipline that challenges the intellectual capabilities of people who are good at algorithmically-oriented problem solving. But a major goal of computer science is to make computers easier to use, and rapid progress is occurring. Today's graphic artists, perhaps with no formal training in computer science, routinely solve problems that were at the frontiers of computer graphics only twenty years ago. The question is who needs to study computer science, and to what level?

Computers in Instruction Trends

Here we will take two approaches, treating each somewhat briefly. In the first approach we will examine an overview of computers in instruction. There the total field is divided into teaching/learning about, using and integrating computers. In the second approach we will look at instructionally-oriented software, instructional support materials and teacher training. There are clear trends in all of these.

Let's begin by dividing the overall field of computers in instruction into three major parts, as illustrated in the diagram.



In the "About Computers" category we include the discipline of computer and information science, of which computer programming is a subfield. The key issue is what do we want students to know about computers. We want to dispel the magic; we want all students to have a mental model of a computer as a machine that can follow a detailed step-by-step set of directions that have been developed by humans. We want all students to have a reasonable level of knowledge about roles of computers in problem solving.

The heart of the matter seems to be the idea of effective procedure--developing and representing procedures designed to accomplish specified tasks. We want students to have some idea about what is involved in a carefully developed specification of a procedure (data representation, data structures and control structures).

We want students to understand roles of computers in problem solving in all disciplines. My current opinion is that we don't do very well in non-computer-based problem solving. Thus, we tend to confuse computer-based problem solving with the more general issue of problem solving. This general issue will continue to beguile educators indefinitely into the future.

It is clear that some form of computer programming can be taught at almost any grade level. However, we lack solid evidence that the teaching of computer programming helps improve general problem solving skills. Also, we now know that if we want students to gain increased skill in writing programs to solve problems, we have to place more emphasis specifically on problem solving. Many people who are currently teaching computer programming courses are not comfortable about significantly increasing the emphasis on problem solving.

My guess is that relatively few students want a "solid" computer science course at the high school level. This, plus informal data I have received from a number of sources, suggests to me that enrollments in straight computer science/computer programming courses at the secondary school level have likely peaked and may well be declining.

Learning and teaching using computers is often called computer-assisted instruction or computer assisted learning (CAL). I believe we are at the start of a steady and significant rate of growth in use of CAL. A number of factors are combining to support this prediction. First, the amount of CAL material is now of significant size and growing quite rapidly. There are several commercially viable companies offering a broad and rapidly growing collection of materials. Second, research evidence to support the effectiveness of CAL is quite strong and continues to grow. Third, declining hardware costs are making CAL more and more cost effective. Fourth, there is an overall educational trend in this country toward individualization of instruction, and CAL supports this trend. Fifth, CAL can increase productivity of students and teachers.

My prediction is that CAL will gradually become a significant factor in the education of most students. I wouldn't be surprised to see that 15-20 years from now more than half of all precollege instruction in this country being delivered in a CAL mode. This suggests a major change in the role of teachers, from delivery of instruction to facilitation of learning.

The integration of computer-as-tool into the curriculum is now well started. Many school districts have set goals of having all students learn to use a word processor, database system and graphics package. The key idea is that such tools increase user productivity--the ability to solve problems. The evidence for increased productivity in the work place is overwhelming. This evidence seems to be motivating schools to change their curricula.

We know, of course, that it is difficult to make major changes in school curricula over a short period of time. The next two decades will be a major challenge between those who understand and appreciate the potential of computer-as-tool and the inertia of our school system. It seems clear that eventually computer-as-tool will be thoroughly integrated into every academic discipline. This will require major changes both in curriculum content and in testing. Students will need access to computers during testing, and this will be a major challenge to the testing industry.

Our second approach to predicting the future of computers in instruction is to look at instructionally-oriented software, instructional support materials and teacher training. The Educational Software Selector, published by EPIE, lists about 8,000 pieces of software. It is clear that the average quality of educational software is growing. The market is now large enough to support a number of companies that specialize in educational software. The competition, as well as excellent leadership in some of these companies, is contributing to improvements in educational software.

Perhaps especially noteworthy is a trend toward development of tools and CAL materials to support longer units or even full courses of study. For example, several companies offer a year long high school geometry course in a CAL mode. The Geometric Supposer is an excellent tool for use throughout a geometry course.

Videodisc-based CAL is gradually growing in importance. A clear trend has been established, with leadership coming from business, industry and military education. Some very high quality materials are now available for use in the precollege curriculum. I predict slow but steady growth in videodisc-based CAL in precollege education.

Instructional support materials include books, films, black line masters, and so on. Conventional publishers understand the production of these types of materials. An indication of how much material is available is provided by last year's state of Texas textbook adoption for junior high school computer literacy. There were 13 books, all copyrighted 1985 or 1986, submitted for their consideration. The computer literacy market is large enough to motivate a number of companies to produce products specifically for it.

The past few years have also seen the publication of a large number of computer-related books to support teacher training. Indeed, this market has been saturated in the past two years, and a number of companies have cancelled some of their planned products in this area. In total, it seems clear that publishers are well aware of the potential for sales of computer-related instructional support materials. Indeed, although there is still plenty of room for high quality materials, it feels like the quality of materials already available is quite large. That is, this is now a maturing segment of the publishing industry.

The teacher training problem is quite large. In the United States there are well over two million inservice teachers, and their average level of computer knowledge and experience is still quite low. Most school districts recognize this problem, and the response of school districts throughout the country has been good. A number of districts have provided all of their teachers with introductory training. The number of certificate programs and master's degree programs in computers-in-education has rapidly expanded in the past three years.

Two things seem clear. First, the average level of teacher computer literacy will continue to grow steadily for many years. Second, this average level of teacher computer literacy will continue to be quite a bit lower than what computer education leaders might desire. Very few current teachers have grown up with computers. We still have very few students entering college who began using computers while they were in elementary school.

Conclusions and Recommendations

Many of the trends discussed in this paper seem quite clear. The hardware price-to-performance ratio for computers will continue to improve quite rapidly. Hardware will be networked. More and better software will become available. Computers will solve or help solve an increasing range of problems. In summary, our access to information and aids to processing this information will increase manyfold in the years to come. Computer usage in government, business, industry and education will continue to grow.

I believe that the educational implications are profound. Several general recommendations are given below. Their full implementation would lead to major changes in our instructional system.

Recommendation 1. Computer-assisted instruction should be viewed as an aid to student-learning productivity. There should be considerably increased emphasis on CAI to make broader educational opportunities available to students, to facilitate more individualization of instruction, and to increase learning.

Recommendation 2. Computer-as-tool should be viewed as an aid to student productivity in making use of their education, both to gain more education and eventually on the job. All instruction at all levels should take into consideration computers as an aid to problem solving and computers as a source of problems. The use of computer-as-tool should be integrated throughout the curriculum. Curriculum content and testing should be modified to adequately and appropriately reflect computer-as-tool.

Recommendation 3. Computer-as-tool should be viewed as an aid to teacher productivity. All teachers should have good access to computers to aid them on the job as well as in their own lifelong learning task. For example, every teacher should have good access to a personal computer at work and at home. Essentially every classroom should have a computer with large screens or a projector to allow computer-based interaction between the teacher and the class.

Recommendation 4. Computers are a severe challenge to our preservice and inservice teacher education program. All teachers will have increasing need both for general computer literacy and for relatively deep knowledge on use of computers within their own specific subject areas. We should place much more emphasis on providing educators with such learning opportunities and facilitating their taking advantage of such opportunities.

Recommendation 5. Computers will change the role of teachers. There will be less demand on teachers to be the source of information and the delivery device. There will be increased demand to be a facilitator--a role model who aids students in acquiring increased "people skills" and high-order thinking and communication skills. All preservice and inservice teachers should be given appropriate opportunities and encouragement to improve their abilities to function well in this changing environment.

Recommendation 6. The totality of accumulated knowledge of the human race is growing very rapidly. Learning to access and make appropriate use of information is at the very core of education. All schools should provide good access to computer-based information systems. All students should be given instruction in use of such systems and should make regular use of these systems throughout their schooling.

Recommendation 7. The ideas of effective procedure (including the creation and representation of procedures, and algorithmic thinking) is among the most important academic ideas of our century. All students should learn these ideas and their impact on various aspects of what it means to be educated for life in our society. Inherent to this recommendation is that students should learn enough of the general capabilities, limitations and underlying nature of computers so that the magic of computers is dispelled.

Recommendation 8. Most real world problems are interdisciplinary in nature. Schools should place increased emphasis on cross fertilization among disciplines, applications of one discipline to the study of a second, and solving problems making use of information and ideas from several disciplines. The computer can help motivate this change in educational emphasis, and it is a valuable tool in such problem solving.

Recommendation 9. Computers will change our world view, our metaphors, our ways of dealing with everyday issues and problems. We should be aware of ways computers are changing our world and not lose sight of important underlying values.

Computer Integrated Schooling

Michael J. Neill

According to some futurists, we have been forging a new society for the past thirty years based upon information, service and electronics. This society employs a growing number of clerks and service workers. It emphasizes a multiplicity of consumer options, decentralization of decision-making, automation of industrial processes and widespread use of electronic information and communication systems.

The industrial and financial sectors have responded to information and communications technology. The modern office is becoming fully automated. Factory processes are controlled by computers or are microprocessor-based. Product design is accomplished by computer-assisted-design (CAD) systems. Components are fabricated on numerically controlled devices and shipped to assembly plants just-in-time by computerized inventory systems. Finished products are assembled by robots. Transportation, accounting, payroll and personnel systems are managed by computer and reports are transferred electronically to the desks of humans who need them.

The implementation of automated functions in any of these industrial processes has resulted in productivity gains -- increased production and/or lower per unit cost. However, industrial managers now see the need to link these "islands of automation" into a coherent system. Such a system will allow any one process to draw immediately upon the data produced by other processes. The "control" or administration of the plant (or industry -- processes could be carried out in geographically dispersed plants) can receive and route information and commands from one "island" to another (see figure 1).

Computers play an important role in each process and in the communication of data to and from control. Industrial technologists are calling this system Computer Integrated Manufacturing (CIM). Productivity gains from integration of automated processes are predicted to be much greater than those attributed to all of the separate processes taken together.

The CIM plant eliminates much of the paper work of the manufacturing processes. Output from a design team, for example, is a digital representation of the design rather than blueprints and paper specifications. This output is the input to the fabrication subsystem. Computers order fabricated parts as inventory decreases. The new parts are shipped to arrive at the assembly point just as the last of the old parts is used. Shipping documents arrive over telephone lines. Paper is produced only where audits are required.

The report of the Carnegie Task Force on Teaching as a Profession cites the need for schools to produce a higher-quality product with greater efficiency. School personnel are faced with the public's perception that they are not very efficient. Student performance over the past decade appears to have declined; the cost of educating a student has increased. Educators are looking for methods which will increase the efficiency of the process we call schooling.

As in business and industry, technology will be a major factor in producing a better educational product and decreasing the unit cost. For twenty-five years schools have been creating their own "islands of automation", some of which have resulted in less administrative overhead and better instruction. District payroll, test scoring, class scheduling, and student record-keeping are examples of systems which have been automated for years. A few attempts to link two or more of these functions have been successful. Inevitably, school administrators will find ways to make schooling more efficient by linking more automated processes. These linkages will make more information available to the people who need it and eliminate much of the paper work. Perhaps such linkages will someday result in Computer Integrated Schooling (see figure 2).

In the short term we can expect to see the automation of more learning processes and the creation of very useful databases for teachers and administrators. Physics lessons, for example, are being developed for interactive video disk players. Several groups have automated the retrieval of curriculum objectives and test items from databases. Agridata, an on-line database designed for the agricultural community, has added files of teaching units and lesson plans which vocational agriculture teachers can access. Bar-coded student identification cards are being used for attendance purposes in New York City schools. The Utah Department of Education is offering Spanish courses to schools which are geographically distant using a mix of satellite communication and computer assisted instruction. Some of these programs will eventually be part of a more comprehensive sytem; others will be good quality technological options for school districts which wish to enhance the productivity of their operation.

It is possible to develop integrated systems from component subsystems. Consider, for example, the typical scheduling and grading cycle in the secondary school. At each step --

course selection, scheduling, production of class lists, mark reporting, generation of permanent student records -- manual data entry and paper output are required. An integrated system would generate a schedule from student-input course requests. Class lists could be output to diskettes in a format which would allow the teacher to maintain them on a gradebook program on a personal computer. Grades turned in on diskette would be input to the reporting system which would generate transcripts electronically and complete the cycle.

Each of these scheduling subsystems has been automated. In Oregon, students are planning their high school programs on a computer system which stores data on career possibilities, career-related subjects and high school graduation requirements as part of the Career Information System package. Teachers everywhere use specialized gradebook programs or generalized spreadsheet programs to manage and report student grades. Scheduling packages have been in use for over twenty years. Controlling the input and output from each subsystem to make an effective package will require time, money and analysis. The resulting system, however, will eliminate much paper processing and data entry -- time resources which can be used for educational tasks.

One elementary school district in Cook County, Illinois has developed an integrated approach to its administrative functions. The Indian Springs District has implemented a computer system which the superintendent claims increases the efficiency of school personnel by eliminating much of the paper work and manual data retrieval that characterizes schooling. The system maintains student, personnel and fiscal records on a medium sized mainframe. Each classroom and administrative office is equipped with a personal computer. The PC can be used as a stand alone for student or teacher use. Communication software is also available to send files or inquiries to the mainframe and to receive back responses. One goal of the system is to eliminate paper except for external communication such as letters to parents or reports to other local districts or to the state or for documents which require signatures.

Educators are also enlisting private sector support to develop more productive human and technological systems. The Center for Excellence in Education at Indiana University is proposing to corporations that they become partners with the university and the Indiana Department of Education in developing models to assist educators to:

-- individualize instruction,
-- manage routine tasks,
-- decrease teacher isolation, and
-- increase student's active learning

The Center will focus on the use of technology and on improving administrative and instructional practices in order to achieve these goals.

The Center is new and funding for specific projects is still being negotiated. However, discussions among university, state of Indiana and industry representatives have begun on how to apply computer technology to school problems. Long term planners envision the development of automated educational tasks which may ultimatly be integrated into a coherent educational program. Such planning includes the development of a computer-based network which will allow communication among personnel from local schools, state agencies and colleges and universities. This network would promote greater sharing of effective classroom and administrative practices. It would serve both as a source of data for educational research and a point of access for research data and results. The network would permit access to databases of test items, course objectives, curriculum units, lesson plans and other resources. In addition to the computer nodes, the Center expects to be a primary user of INTELENET (Indiana Telecommunication Network), an "electronic highway" which is expected to link Indiana's 92 counties by early 1989. INTELENET will make any developments on the network accessible to most Indiana educators.

The Indiana Department of Education has looked at ways to individualize instruction and to disseminate developments to schools throughout the state. Staff members of the Department have expressed a need for a computer-based system whereby state proficiency criteria for all levels and subjects can be customized for students. These criteria will form the basis for individualized educational programs for students to be developed at the local district level. Educators visualize that resulting educational plans, student achievement information and other relevant data would go back into the database for analysis and possible refinement of the criteria.

These examples are but a small number of possible developments in the application of technology to educational efficiency. The prospects are exciting; the problems are obvious. However, the problems involved in automating schooling are no more formidable than those experienced by other major industries during the past decade. Staff acceptance, data and communication compatability and capital costs are common problems. Even conceptualizing what a total system should look like will require considerable planning and analysis. Solutions will include staff training, new hardware and software packages, new management techniques and improved organizational models.

Educators are examining the role of computers to enhance educational productivity at the same time they face the need for new curricula and a shortage of trained teachers to implement them. These additional problems will undoubtedly require technological solutions and therefore should be viewed as an opportunity to think about education in novel ways. Technology will not be readily accepted unless it truly helps educators do better work. There is every indication that current developments in educational technology will do exactly that.

Dr. Michael Neill is the director of the Indiana Clearinghouse for Computer Education which is located on the Indianapolis campus of Indiana University

INTEGRATING
MANUFACTURING PROCESSES

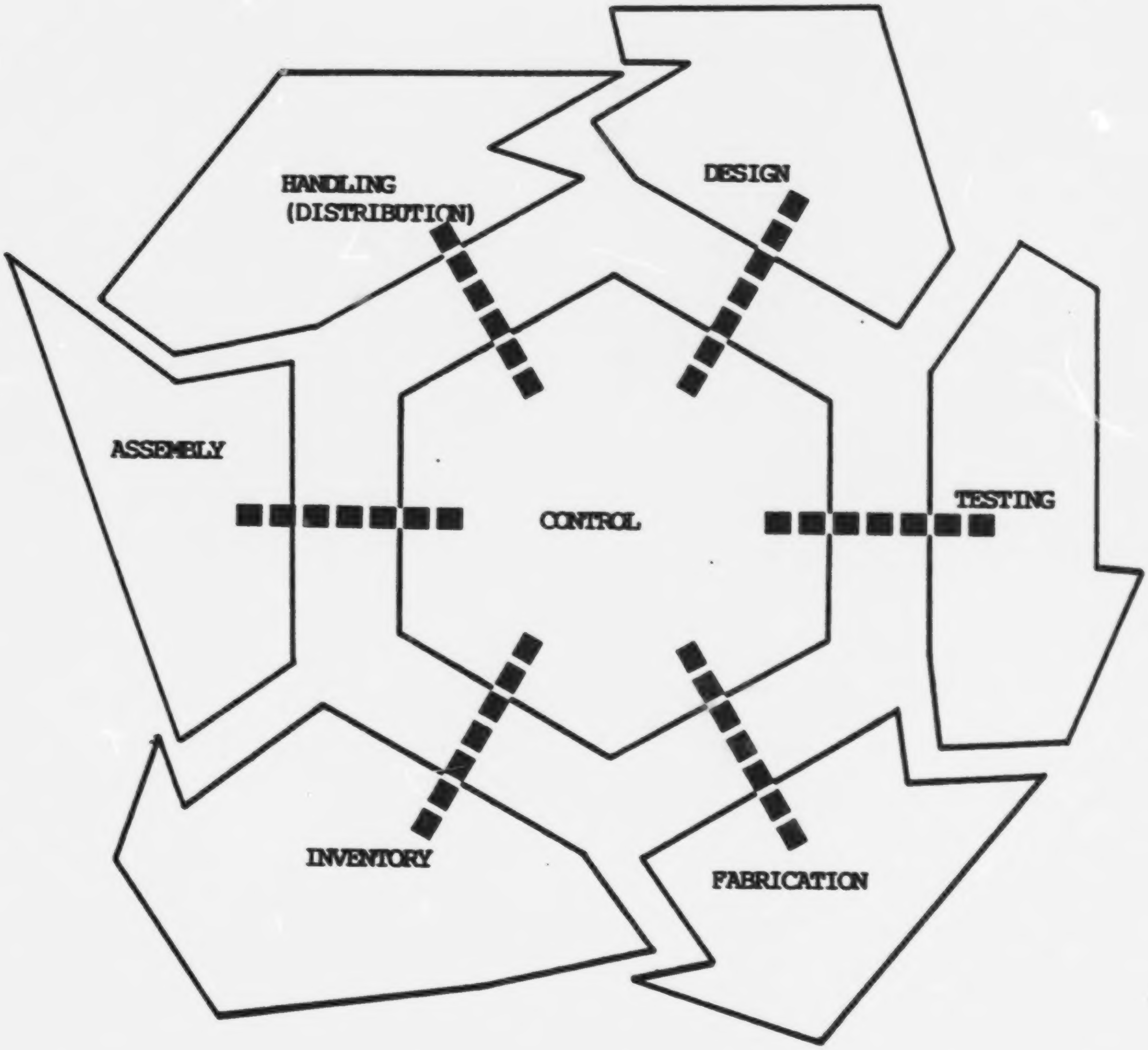

HANDLING
(DISTRIBUTION)
DESIGN
ASSEMBLY
CONTROL
TESTING
INVENTORY
FABRICATION

FIGURE I

INTEGRATING
EDUCATIONAL PROCESSES

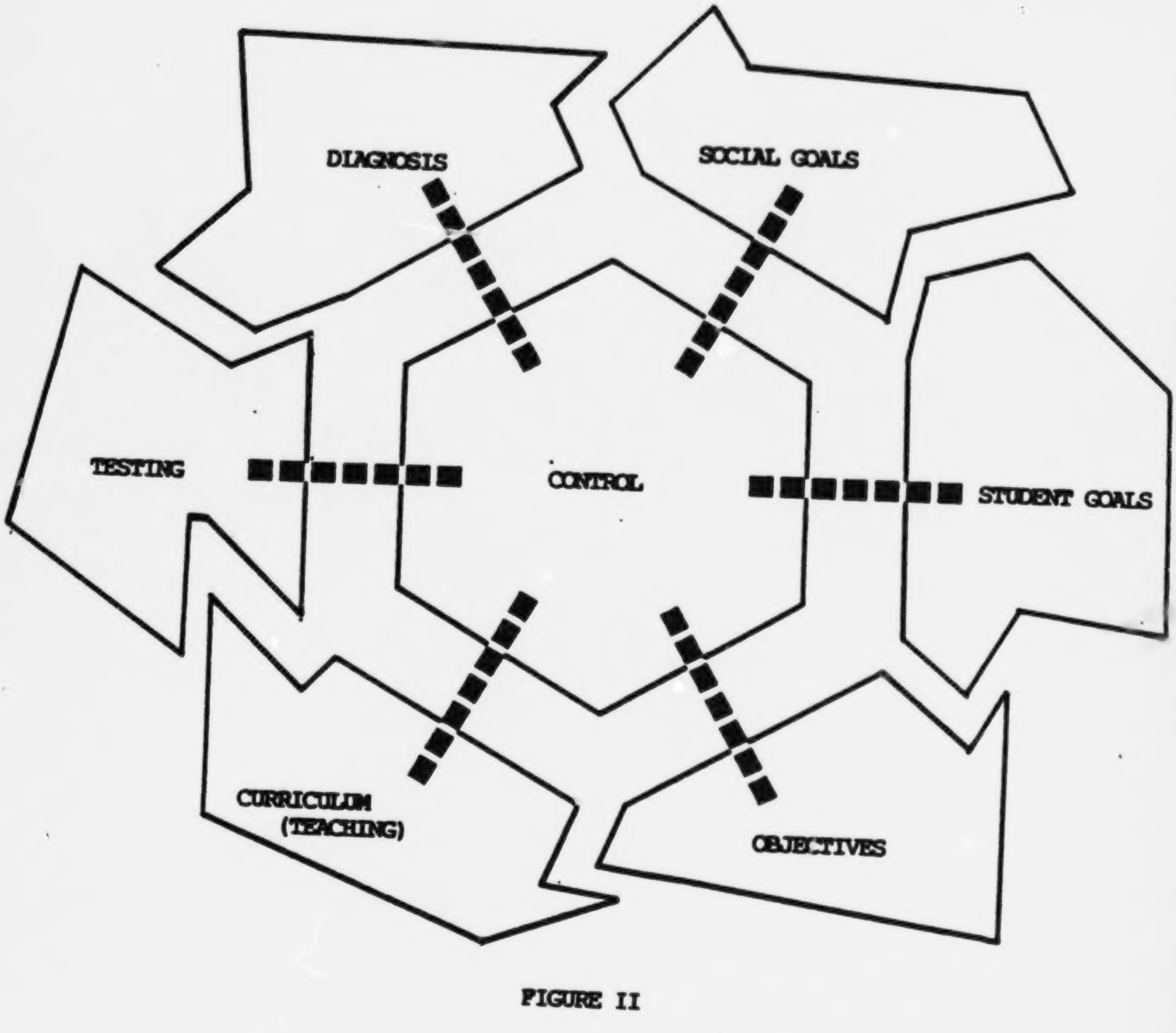

DIAGNOSIS
SOCIAL GOALS
TESTING
CONTROL
STUDENT GOALS
CURRICULUM
(TEACHING)
OBJECTIVES

FIGURE II

NEW DIRECTIONS FOR TEACHING BASIC PROGRAMMING

Andrew M. Paulson

Man learns by looking to his past and evaluating his successes and failures. In searching for new directions in teaching BASIC programming, I felt the need to do just that. I began looking back and this is some of what I was able to uncover.

Beginnings of High-level Languages

The first high-level language of lasting significance was FORTRAN (FORmula TRANslator). The language was first used on an IBM computer in 1956. The project leader that developed FORTRAN was John Backus of IBM. He later participated in the development of a formal method for defining the syntax of programming languages which became known as the Backus-Naur Form (BNF).

In 1958 the language ALGOL (ALGOrithmic Language) was developed. Like FORTRAN, ALGOL was effective in solving a wide variety of mathematical problems. However ALGOL, unlike FORTRAN was not actively supported by hardware manufacturers in the U.S. Therefore it was much more popular in Europe.

The U.S. Department of Defense, in 1959, undertook the development of a common language for businesses working under contract to the department and other governmental agencies. The result was COBOL (COmmon Business Oriented Language). Unlike FORTRAN and ALGOL, COBOL emphasized readability of programs; a COBOL program can often be read and interpreted by a non-programmer. COBOL programs are largely self documenting, making them easy to revise and update.

1965 saw the introduction of two significant new languages. One was PL/I (Programming Language i). PL/I was developed under the direction of IBM and was an attempt to combine the best features of ALGOL, FORTRAN and COBOL. It was predicted that PL/I would become the dominant high-level language and replace the others. Obviously this did not happen. One reason was that existing COBOL and FORTRAN users didn't feel the additional features of PL/I justified converting to the new language. Another problem was the "size" of PL/I; it has numerous facilities and statements and takes much longer to learn than COBOL and FORTRAN.

The second language introduced in 1965 was developed at Dartmouth College. Professors Thomas Kurtz and John Kemeny developed a teaching language loosely based on FORTRAN and ALGOL. Their new language is known as BASIC (Beginners All-purpose Symbolic Instruction Code). BASIC has strong interactive facilities: it is possible for a user to "converse" with a BASIC program as the program is executed. This makes it possible for the program to "ask" the user to supply data, verify data, make choices, etc., as the program is executed. BASIC's exploded with the introduction of microcomputers beginning in 1975. Today BASIC may well be the most widely known and used high-level language in the world.

Professor Niklaus Wirth of the Technical University of Zurich, Switzerland developed a significant new language in 1971 called Pascal (after Blaise Pascal, a well-known seventeenth century mathematician). Pascal was based on ALGOL but incorporated several refinements. Pascal's ability to manipulate non-numeric data is far superior to ALGOL. It has superior input and output features. Data structures, such as lists and tables, can be easily manipulated and it is possible for the users to define and use their own data types. Pascal, like ALGOL, is a block structured language. this means that programs are divided into sections, or blocks, to accomplish specific functions and that alterations can be made to one block without affecting the others.

A recent language, which promises to have great impact is Ada. It was developed in 1979 for th eU.S. Department of Defense as a replacement for the numerous languages used by the American armed forces and their suppliers. Ada is based on Pascal but includes several extensions and new features. Learning Ada is a major task for any programmer due to its size. It is especially suited for real-time applications.

Evolution of a Language

The Birth Of BASIC

In the early 1960's, computers, in the modern sense had existed for about ten years. These computers were very large and expensive beasts. These beasts were housed in very light, cool and sterile environments referred to as "the computer center". These rooms were designed to protect the machines from the incompetencies of human kind.

The human user punched a program on IBM cards and submitted it to an operator. The operator collected hundreds of such jobs and fed them to a computer in a "batch". The machine worked on one task at a time until it was completed, then printed results on a not-so-fast printer. When all jobs in a batch were completed, it was time for the next feeding of the beast.

The user returned the next day to pick up the results. This seemed a reasonable wait based upon the fact that some of the tasks could take humans weeks or months to complete. Thus "batch processing" was universally accepted.

It might have been a good system if the user had received a solution the next day. But what we found on our printouts were error messages like "Illegal instruction on card 123". Then the long process of "debugging", or finding errors, began. The first few days were needed to correct only keypunch errors.

These early frustrations led Professors Kemeny and Kurtz to do some real brainstorming when designing a computer system for Dartmouth College in the early 1960's. They decided to create a "really nice language" to use on a "time-sharing" system. Until that time, neither had been developed.

Early experiments at MIT and Bell Labs were designed to allow a number of people to use the same computer at the same time (time-sharing). the great speed of the experimental computer would allow them to receive fast service. A program that used to take days to debug using the batch processing system, should eventually take fifteen minutes to half an hour using the new system.

BASIC was the offspring of these early experiences and frustrations. It was strongly influenced by FORTRAN and ALGOL.

They had some definite ideas of what features their language should have:

1. It should be easy to learn for the beginner.

2. It should be a general-purpose language, allowing the writing of any program.
3. Advanced features had to be added so that, if there was a price, it was paid by the expert, not the novice.
4. It should take full advantage of the fact that the user could interact with the computer.
5. It should give error messages that were clear and friendly to the user.
6. It should give fast response for small programs.
7. No understanding of the hardware should be necessary.
8. It should shield the user from the operating system.

Much of the design of BASIC was dictated by these eight principals.

To use this new language a BASIC compiler designed by Professor Kemeny.

Then on May 1, 1964 at four o'clock in the morning, John Kemeny and a student programmer were sitting at neighboring terminals connected to the same computer. They were typing simple programs in BASIC. They typed RUN at the same time and were deliriously happy when they both got correct answers. Time-sharing and BASIC were born. Thus interactive computing was made possible by a combination of BASIC and time-sharing.

Adolescence and Maturity

The development of BASIC worldwide suffered from having "too many cooks". However, at Dartmouth, the eight original principals kept researchers pointed in the correct direction.

From 1964 to 1971 BASIC underwent some major changes and revisions in its developmental processes. In 1964 the language contained these statements:
LET, PRINT, END, FOR, NEXT, GOTO, IF-THEN,
READ, DATA, DIM, DEF, GOSUB, RETURN, and REM.

During this period we saw the development of numerous statements and facilities into BASIC. These included: INPUT, PRINT USING, MAT, default values, string processing, data files, subroutines, and indentation.

In September of 1971 the "switchover" to the new version occurred with no ill effects. It served as the principal version of BASIC at Dartmouth until 1979, and as a subsidiary version even to today.

By 1971 every effort was made to have computers run very fast, to permit large numbers of students to simultaneously code, compile and test in the system. It included an online debugger. It also allowed large programs by automatically overlaying subprograms no longer needed with ones that were needed. These subprograms could call themselves recursively. All in all BASIC was a huge success and we were all ready for the seventies.

Maturity

Through the seventies and into the eighties technology advanced beyond the wildest dreams of those involved in the primal development of the languages of computers.

in the late sixties and early seventies a famous computer scientist, Edsgar Dijkstra, was developing the science of structured programming. He pointed out that the difficulty of testing and modifying a program seemed to be directly proportional to the number of GOTO statements it contained. His work and comments could not be ignored. However, Structured Programming is one of the major memory glitches of the developers of BASIC for the microcomputer.

The 1970's also saw the introduction of the CRT to replace the teletype terminals. This gave rise to the revolution of plotting and graphics which, previously, could only be done on very expensive research computers with flat-bed plotters.

In 1974 the American National Standard Committee (X3J2) was formed to produce a standard for Minimal BASIC, which received official approval in 1978. X3J2 then set its sights on full BASIC. But by then it had to cope with a "moving target". The explosion of personal computers, structured programming and graphics. It took far longer to devise the standard for the full language.

The Future

Portability is a main concern of most BASIC users. That is, the ability of all machines to use the same code.

For years now, we instructors and users have been searching for the "ideal" version of the language to suit our needs as well as those of our students. Basing our opinions on past exposure and the directions we and they should take in the future.

To make intelligent decisions about the future, we need to look at the past and the present.

What went wrong? Why are there so many versions of the one language, BASIC?

Many versions were written to accommodate "tiny" computers (those with very little memory) so severe compromises had to be made. As memory expanded, larger more powerful versions of BASIC were demanded and produced, but they were based on earlier versions. Compromises became features of the new "street BASIC" language.

Manufacturers were anxious to take full advantage of the hardware they had available. So they added hardware dependent features. The result has been incompatibility. Programmers don't write in BASIC but instead they they use "BASIC for the Apple" or "BASIC for the IBM-PC" or "Commodore BASIC". These have come to be referred to as "Street BASIC".

The blame lies in several other areas also. aside from street BASIC, the BASIC interpreter is a major culprit. Error detection is done through the interpreter. The construction and design of interpreters has, in the most part, come out of the fact that one is short of time or computer memory or both.

Because of these limitations on time and memory, looping, in some versions, has become very awkward.

We can discuss graphics, at length,and arrive back at the same place. There is no reasonable explanation not to adopt a common language which can accomplish the desired graphics task using the same code.

The problem, then as I see it, is one of lack of standardization. Standardization is one sure way to prevent the degeneration of the language into street BASIC.

What happened? Two things. One is that standardization did not begin until 1974, ten years after the invention of BASIC.

The second reason is that, once the standards process had begun, the rules changed. By the time the standard for Minimal BASIC appeared in print (1978), it was obsolete. In the early 1980's, it became obvious that computer memory was very cheap and that language limitations due to small memories were gone before any standard appeared.

Where are we headed? The draft proposed American National Standard for BASIC (dbANS) is a standard for the language. It describes the rules that must be followed in writing a program if that program is to be standard-conforming.

The crude "street" versions that high school and college students all over the country have to use must become standardized.

Professors Kurtz and Kemeny have accepted the task as a challenge to build a BASIC compiler that would run on just about any brand of personal computer. This strategy has worked for the Pascal language.

They are developing the True BASIC System. The heart of the system is the BASIC compiler. It is one of the first compilers to be based on the dpANS standard. The BASIC language it uses is extremely close to the Standard BASIC, but there are differences.

So now what do we do in the interim? We maintain a high level of consistency and quality and demand the same of our students. We teach a block structured BASIC even if it is resident to the machines at hand.

I have attempted to locate Pseudo languages which assist the students in their progression through their exposure to programming languages.

Their experiences usually begin with LOGO, progress through BASIC and into Pascal at the high school level. Both LOGO and Pascal are structured languages where you define procedures which can be called from within the program. I have located a pseudo BASIC which has the same or similar structures. I teach a project oriented curriculum to enable the students to progress through the languages at a comfortable pace.

It is my hope that Professors Kemeny and Kurtz's work on the development of True BASIC proceeds as planned. Currently, True BASIC is available for the IBM-PC, the Apple Macintosh and the Commodore Amiga, offering true portability between these machines. I hope that their work expands into the areas of the other major microcomputers currently being used in education, and that future hardware developers recognize the importance of standardization in BASIC programming. This standardization of the language will, inevitably, make BASIC programming easier and more enjoyable to teach, learn and use.

References

Chandor, Anthony, The Facts on File Dictionary of Microcomputers. New York: Facts on File Inc. 1981. 184 pages.

Helms, Harry, The McGraw-Hill Computer Handbook. New York, et al: The McGraw-Hill Book Company 1983. 992 pages.

Kemeny, John G., and Kurtz, Thomas E., Back to BASIC. Reading, Mass., et al: Addison Wesley Publishing Company, Inc. 1985. 141 pages.

Nelson, Dale, et al, BASIC A Simplified Structured Approach. Reston, Va: Reston Publishing Company Inc. 1985. 232 pages.

Rhengold, Howard, Tools For Thought. New York: Simon and Schuster 1985. 336 pages.

Stire, G. Harry, The Untold Story of the Computer Revolution. New York: Arbor House 1985. 219 pages.

Thornburg, David D., "With Friends Like BASIC, Who Needs Enemies?" A+. 4, 2 (February 1986), pp.79-80.

IDEAL USES OF THE COMPUTER IN SCIENCE

Bill Sherzer

As a science teacher I spend a lot of time trying to change students perceptions of science. The belief of many people is that certain things are science. I try to change this belief into an understanding that science is a process. What is the science process? Simply stated the science process is: observation, hypothesis, experiment, rehypothesize, retest and so on. To focus on one quality of this process as the most 'scientific' I would find that experimenting is the key element in science.

There are a bewildering variety of programs available to teachers. There are drill-and-practice, tutorial, simulations, diagnostic, utility and of course games, to name a few. How do you pick out the program to recommend for purchase in your classroom? Before jumping in on the purchase of interesting sounding programs what do you want a program to do? Tutorials are OK, but I have troubles justifying to myself the expense of both software and especially hardware for an approach that can be duplicated far more cheaply with a class set of pamphlets. How do you justify a computer program? Refer back to the end of the first paragraph; I want a program that allows a student to experiment. If a student can't experiment I have troubles in seeing how I can call the process science.

Some of my ideals in a science program would include:

1. Visual display - a program should use the graphic and computational abilities of the computer. This should be a real enticement, an alternative to the static pictures of the chalkboard or overhead.

2. Experiment - the user should be able to try a number (or condition) and see what happens. This is a step in gaining experience.

3. Simple to use - menu selection is a must! If spelling is not critical, then don't make spelling critical. Often typing just slows the creative process of students down. We want creativity! As much as possible, the commands should be highly mnemonic, easy to remember.

4. Open-ended - Once the basics have been mastered you (and hopefully the students) should feel that you could try a variety of things with the program.

5. Inexpensive - I think this is self-explanatory.

To sum up: a program should not be solved or have an end to it. The program should be a tool. Once learned, the software should aid you in understanding our world a little better. You should want to experiment! With "The Observatory" I've had as many as four computers working on different set-ups because I couldn't wait. That is exciting! That's the kind of program I want.

I will be demonstrating three exciting programs to you: Broderbund's "Science Toolkit", Vernier's "Charged Particles II" and Lightspeed's "The Observatory". These programs are from different distributors, involve different topics and do quite different things, but they can all be used as tools.

Science Toolkit
Broderbund 1985; Apple II+/IIe/IIc with 64K; $59.95

Description-

The Toolkit includes a thermistor, photocell, interface box and connecting cables in addition to an excellent menu driven program. The program allows the user to configure the computer to behave as a thermometer, a lightmeter, a timer or a stripchart recorder. The menu looks like a notebook with marked flip tabs to allow fast reference and to emphasize to students to use their notebooks. From the menu, the tools can be selected, data saved or retrieved from disk, print data, and set-up parameters for various tools.

Ideal-

The graphics are excellent with the "Science Toolkit". There is no doubt when your computer is a thermometer because a large thermometer is vividly displayed with a colored column marking the temperature. In addition, the information is given digitally beside the thermometer. A similar display is given for the light meter. In addition to the graphics, audio is included for even more attention getting.

The excellent user's manual includes twenty-seven experiments for the student and additional experiments would be easy to invent. Very obviously, this program is a tool. Some of the experiments include: Radiators and Elephant Ears, What Stardust and a Dirty Window have in Common, Nature's Helicopters, Chemical Refrigerators, Building a Better Ski Jump, etc.

The style of menu with flip tab notebook appearance makes it very easy to select and remember what step you are on. The constant emphasis on a notebook is a great reminder for students to keep notes. Cues are constantly given at the top of the screen once you are in an experiment so the user is not in doubt as to what should be done.

The program is obviously open-ended. The "Science Toolkit" is about as limited as a thermometer or a photometer would be. The possibilities for experimentation are unlimited.

Demonstration-

The demonstration will not be a fancy and potentially non-working example. The ease of set-up and a simple temperature change will be displayed.

Additional Ideas-

The user's manual has an extensive list of ideas. Further probes are expected in the future to open even more avenues for exploration.

Charged Particles II
Vernier Software; Apple II+/IIe with 48K; $24.95

Description-

Vernier has a whole line of physics related software. Their programs include: Orbit II, Ray Tracer, Frequency Meter, Kinematics II, Wave Addition II, Precision Timer, Vector Addition II, Graphical Analysis II and this one on charged particles. After selecting a challenge from the menu, you are given a grid display. You control three variables; the magnetic field strength, the electric field strength, and the injection speed of the particle. The various challenges include: making an electron orbit with a 5 cm radius, making an alpha particle orbit, determine the e/m ratio, determine the mass of an electron, construct a velocity filter, use the filter as a mass spectrometer. By varying the field strengths and particle velocity you watch your results on the grid display and vary your next test accordingly.

Ideal-

The visual display is adequate and actually gets very exciting if you leave several traces on you screen (erasing or leaving you traces is one of the options at each test). After setting up each set of parameters you get to view the trace develop. This is very nice and gives a sense of a real experiment; a real happening. Within each challenge you can vary the fields as you with and view the results.

After getting used to "computer scientific notation" (0.0001 = 1 E -04) the field strengths become much clearer. Though you could change the program, (more about this in a moment), you can not put in any value for field strengths but are restricted to step increments. Changing the field strengths is easy to remember as the physics abbreviations are used (E = Electric field, B = Magnetic field). Menu selection is used for picking challenges.

Vernier's approach to the application of their programs to science is terrific. First, they have set up these challenges to get people motivated. You want to solve the challenges! There is no right or wrong with them. You will not win a prize or have the computer throw a fit if you win. You judge if you have solved the challenge - very much like the real world. Second, and most important, Vernier publishes their program in an open fashion. You may list and modify their program. You can change the value of the unknown particle "Z", you can change the title or set-up of the display if you wish. Vernier even offers a "Modification You Might Want to Try" in their documentation. All Vernier asks is that if you come up with a great idea drop them a line so that they can use it in their next edition.

Demonstration-

The demonstration will focus on Challenge 1 (Orbiting Electron). Several field changes will be made and the effects on the electron observed.

Additional Ideas-

Once the various challenges have been tried the experimenter is free to use the set up as a mass spectrometer or repeat some of the famous experiments in the past, e.g. Thompson's e/m ratio, Milliken's oildrop. Follow the suggestions made for modification #1 to change the unknown particle to any mass/charge. Then you can have class contests and see which student or group is best at interpreting a mass spectrometer.

The Observatory

Lightspeed Software 1984; Apple II+/IIe with 16K RAM card; $39.95

Description-

After booting up, you see a sky full of stars. The program is a planetarium. You can pick any spot on the earth (by entering latitude and longitude), on any date 10,000 years in the past or future. You may magnify one section of the sky (up to X256), draw the constellations in and even ask what object you are viewing. The sun, the moon, even Venus and Jupiter can be expanded sufficiently to be viewed as disks. Over 300 objects can be found including galaxies, nebulae, all 9 planets and even the four Galilean moons around Jupiter. The moon can be seen it its phases. A time interval option allows you to advance or back up by seconds or years. The neat things this program can do just goes on and on.

Ideal-

This program is without a doubt one of the best computer programs around. The visual display is mostly astronomical objects with the left side displaying necessary printed material. The ability to change times, locations, zoom in, draw constellations for reference gives the user great latitude in trying "what if I . . ." Commands are single, mnemonic keystrokes, e.g. Z - zoom in or out, C - draw or undraw constellations, T - allows time changes. Selection of viewable objects is from a "perforated menu"; a really innovative feature. I find this program better than a real telescope because I can compress years of sightings into minutes. Weather and late night hours don't get in the way. Basically, I set up a problem and spend my time experimenting. Even if the price were to double, the cost in very reasonable.

Demonstration-

I will boot up with the proper time and location.

After displaying the constellations, I will find Jupiter and zoom in and observe the moons. I will also briefly show a solar eclipse.

Additional Ideas-

My biggest introductory classroom success has been the "Planetary Treasure Hunt". I set up teams, list the planets and let the kids go to it. Talk about excitement! Expanding on this idea, I have several treasure hunts for all grades now, form first to adult.

Some experiments to set up are: how long does it take for ------ to orbit? (the Galilean moons, the planets, Halley's comet), where is the zodiacal plane?, what is the relationship between the moon's phases and the time it rises?, what does the orbital path of a planet look like?, what does the orbit of Halley's comet look like?, and many more.

This year for the annual Science Fair, two students came up with projects using "The Observatory". One tried and succeeded in finding the transit of Venus in 2004 A.D. The other was a group of two girls who came up with the length of Jupiter's year. They came up with a value of 10 Earth years instead of the listed 12 years; but they have a lot of respect for astronomers now!

THE ROLE OF SCHOOL-BASED COMPUTER COORDINATORS AS CHANGE AGENTS IN ELEMENTARY SCHOOL PROGRAMS

by Neal Strudler

School districts are committing large amounts of money to teacher training, equipment, and instructional materials in the hope that computers will become an effective component of educational programs. To facilitate the implementation of this innovation, some schools have created a new role: the school-based computer coordinator. Little is known, however, regarding the specifics of that role. What do computer coordinators do? What strategies do they employ? What skills do they need to be effective? This paper offers a summary of a research study designed to answer these and other related questions.

Background for the Study

Since computer use in schools is a relatively new phenomenon, much of the present literature describes "model" efforts, ideas for curriculum innovation, and plans for district-wide change. Few studies have systematically documented the actual transfer of such ideas and training into widespread classroom practice. There is much that we can learn, however, from studying the "track record" of previous educational innovations. Thus, it is helpful to consider what researchers have learned about educational change in general and to build upon these findings towards a new line of research regarding the implementation of computers in schools.

The Need for Teacher Training

The Rand Change Agent Study (Berman & McLaughlin, 1977) found that even the best innovation could not succeed with inadequately trained or uncommitted teachers. Staff development, therefore, is critical for the implementation of new programs and teaching methods. One element of effective staff development is support following initial training (Gall & Renchler, 1985). Follow-up support helps teachers to transfer newly learned teaching skills and practices into their active teaching repertoire (Joyce & Showers, 1983). The present study examines the computer coordinator's role in facilitating the transfer of teacher training with regard to instructional computing.

Arthur Luehrmann (1984) and other proponents of computer literacy instruction have argued that computers can be most efficiently implemented through specialized classes with a limited number of specially trained teachers. Where the goal is implementation of computers throughout the curriculum, however, training demands are greatly increased (Moursund, 1985). The present study focuses on the training demands that are involved when the goal is integration of computers throughout the curriculum.

The Role of Instructional Computer Coordinator

A growing number of people hold the position of instructional computer coordinator at the school level (Barbour, 1985; Moursund, 1985). A recent survey in Electronic Learning found the following regarding school-based computer coordinators:

1) Job descriptions vary greatly.

2) Only 21 percent of the repondents actually hold the title "computer coordinator"; many others function in that role on a **de facto** basis.

3) 80 percent of school computer coordinators who responded fulfill their role as an additional responsibility; only 4 percent fulfill their role on a full-time basis while 16 percent function on a part-time or "released" basis.

The present study attempts a detailed description of the computer coordinator's role. It focuses on the emerging group of coordinators who have been provided released time to facilitate computer implementation in the schools.

The Change Agent Role

A recent study of school improvement found that the successful implementation of innovative programs is dependent upon the intervention of key personnel involved in various change agent roles (Huberman & Crandall, 1982). A change agent is defined as an assistance person whose role includes responsibility for facilitating change. This emerging role is being given increased attention because of a shift in the approach to school improvement . Whereas the post-Sputnik school reform campaign tended to rely on centralized curriculum changes, current efforts focus on school-based change. With this change there has been a corresponding shift in change agent roles from a disseminator of ideas and materials to a process consultant (Beaton, 1985).

While change agents are widely used as "special assisters" in current school improvement programs, little is known about their functions, the strategies they employ, the specific skills most crucial for success, or how to train for such skills (Fullan, 1981; Miles et. al., 1985). The Patterns of Successful Assistance study (Miles et. al., 1985) focused on these concerns in its two-year investigation of 17 change agents working in three urban school improvement programs. A synthesis of findings resulted in 18 key skills for educational change agents. Beaton (1985) replicated aspects of the Patterns of Successful Assistance study in focusing on the role of staff development specialists in three school improvement programs. He found that these specialists use a combination of product- and client-centered strategies that require sophisticated interpersonal skills. The main teacher outcomes he reported were motivation, positive attitudes, and enhanced communication and cooperation between teachers.

The present study seeks to extend the findings of Miles and his colleagues and Beaton. It was designed to replicate aspects of their studies to investigate a new change agent role: the school-based computer coordinator.

Significance of Study

Knowledge of the computer coordinator's role may help school districts and other planners to formulate policies and procedures that guide allocation of these personnel. Information on successful strategies used by computer coordinators might increase their effectiveness as change agents and enable their supervisors to provide better guidance and support. Furthermore, identification of the skills required in that role could aid in the selection and training of prospective coordinators. Ultimately, knowledge of the role of school-based computer coordinators may increase the likelihood that resources committed to instructional computing will lead to successful implementation.

The Setting: Eugene District 4J

In 1984, the District 4J Computer Council recognized the need for a uniform approach to computer education at the elementary level. In June of that year, a group of teachers, under the direction of the District Computer Coordinator, began developing a program that would focus on integrating the computer into the established curriculum and on providing articulation between grade levels. The program was piloted during the 1984-85 school year. The pilot study was evaluated (Ames, Gilberstad, Sky & Strudler, 1985) and the program was deemed feasible contingent upon continued district support with regard to computer resources and staff development. The evaluators recommended that part-time "computer persons" be designated to coordinate training, maintenance, and scheduling at the school level.

During the 1984-85 school year, only one of Eugene's 24 elementary schools employed a released-time computer coordinator (defined as any coordinator who is assigned .1 FTE or greater for computer-related responsibilities). By the following school year, nine elementary school had opted for released coordinators.

Research Design

This is an exploratory case study that uses questionnaire, interview, and observation procedures to survey three change agents carrying out a school-level computer coordinator role. Fifty two interviews were conducted to gather data from coordinators, their supervisors (principals and the District Computer Coordinator) and their clients (the teachers and parents in their school). Questionnaires were also completed by coordinators and supervisors. In addition, representative activities were observed and relevant documents examined.

Sample

Eugene District 4J currently has nine elementary schools with released-time computer coordinators. A sample of three of these schools was selected for this study. Coordinators were chosen from schools that have been identified as having a high degree of implementation of instructional computing. Additionally, consideration was given to choosing coordinators who vary with regard to critical variables, namely: (1) technical expertise in computing, (2) role expectations of computer coordinators by supervisors, teachers, and themselves, (3) previous teaching experience in the particular school, and (4) available resources.

Data Analysis

All 52 interviews were tape-recorded. Notes and verbatim transcriptions were obtained from each recording. Field notes and interview data are presently being coded for analysis according to the list of constructs developed by Miles. Data will then be analyzed case by case, and then across cases. Finally, the results will be compared with those of Miles and his colleagues (1985) and Beaton (1985).

Preliminary Findings

Although the procedures for formal data analysis have not been completed, I would like to offer the following preliminary findings based on an informal review of the data gathered:

1. The role of the computer coordinator is multi-faceted and difficult. Key functions that coordinators fulfill include:

—securing hardware and software through allocated funds, grant writing, and fund raising activities.
—ordering, organizing, and managing all computer resources
—modeling effective teaching.
—serving as a resource person for choosing software and planning instructional computer use.
—training teachers; conducting workshops.
—recruiting and training parent volunteers.
—facilitating school-wide planning in conjunction with building administrators and the school's computer committee.

2. Effective coordinators display a variety of styles, skills, and strategies. All informants, though, seemed to prefer a facilitative style, characterized by a collaborative, supportive, and collegial approach. This style seems to be especially effective when complemented

by good administrative and organizational skills. Though it involves a difficult balance, effective coordinators seem to combine being patient and "low key" with being active and systematic.

3. Nearly all informants expressed that interpersonal skills for coordinators are more important than their knowledge of computing. Supervisors added that "people" skills are more difficult to teach and learn than are computing skills. Thus, they felt that people chosen to be coordinators should already possess strong interpersonal skills.

4. Effective coordinators take into account the "plight" of the classroom teacher. Elementary teachers are overloaded with the demands of a crowded curriculum. Coordinators can avoid adding further responsibilities by helping to screen for high quality, appropriate software. In addition, coordinators can ease the demands on teachers by being responsible for having computer labs "pre-booted" and ready for student use. Additional time-savers might include helping to print student work (especially slow printing graphics), put away programs, and provide students with individual attention when necessary. Coordinators can facilitate this assistance by organizing parent volunteers to help in the lab. With less "down time" for students, teachers seem much less resistant to using computers in the curriculum.

5. The role of computer coordinators will likely change with the level of computer use and teacher expertise in a particular school. When teacher comfort with computers is low and resistance is high, a coordinator must "win over" teachers with effective "modeling" of classroom instruction and an encouraging but patient style. A coordinator's teaching skills and "bedside manner" may become less critical as teachers become more comfortable with educational computing.

6. Coordinators face an inevitable conflict between overcoming teacher resistance and facilitating teacher independence . Basically, the more a coordinator does to make things easy for teachers (eg., choosing materials, planning lessons, teaching the lessons, always being there to "bail people out," etc;), the less stress and resistance there tends to be. On the other hand, the more a coordinator provides such help, the more the teachers tend to be dependent on that "helping hand." Thus, the question arises regarding how helpful to be. Ambitious plans for getting teachers to independently "fly on their own" may result in increased teacher competence or greater teacher resistance. Coordinators must judiciously consider the comfort level and commitment of the staff and administration in choosing a preferred time-line and approach for computer implementation.

7. A computer coordinator's goal of working himself or herself out of a job is admirable, though in some cases overly optimistic. While the training and "hand-holding" responsibilities will invevitably decrease in time, coordinators will likely continue to serve as "filters" for the barrage of new technology and software. Even if teachers have the skills to sort through new developments, it is unlikely that the

majority of them will have the time or the interest that it takes to remain current.

8. It appears to be an advantage to have a coordinator who has been chosen from the existing staff of a school. With a sense of rapport and trust in place, that coordinator may be able to be more ambitious in moving the program along. A coordinator chosen from outside the staff has a heightened need for the interpersonal skills to effectively collaborate with a new faculty. If that staff is committed to expanded computer implementation, however, then a skilled coordinator may experience few, if any, obstacles in adjusting to the new school situation.

9. A .5 FTE seems adequate for the computer coordinator of a medium-to-large sized elementary school. If less time is allocated, role expectations should be adjusted accordingly.

10. Outcomes realized due to the work of released-time computer coordinators include: the improved skills of teachers and the readiness for further advances, the implementation of program goals, teacher satisfaction with the program, and increased feelings of self-esteem and professional growth.

Implications for Practice

The present findings suggest that school-based computer coordinators perform a variety of functions that facilitate computer use in schools. If integration of computers throughout the curriculum is a school-wide goal, then a released-time coordinator should definitely be considered. Increased computer implementation involves many demands and responsibilities that clearly need to be coordinated. While prospective coordinators should be well-versed in their knowledge of instructional computing, it appears critical that they possess strong interpersonal and organizational skills. As with previous innovations, successful computer implementation requires effective planning, training, and support, all of which can be facilitated at the school-level by skilled coordinators.

References

Ames, C., Gilberstad, N., Sky, N. C., & Strudler, N. (1985). *Elementary Computer Program Pilot Study Evaluation*. Unpublished manuscript, Eugene School District 4J, Eugene.

Barbour, A. (1986, February). "Computer Coordinator Survey." *Electronic Learning*, pp. 35-38.

Beaton, C. R. (1985). *Identifying Change Agent Strategies, Skills, and Outcomes: The Case of District-Based Staff Development Specialists*. Unpublished Doctoral Dissertation, University of Oregon, Eugene.

Berman, P. & McLaughlin, M. (1977). *Federal Programs Supporting Educational Change, Vol. VIII, Implementing and Sustaining Innovations*. Santa Monica, Calif. : Rand Corporation.

Fullan, M. (1981). "School District and School Personnel in Knowledge Utilization." In R. Lehming & M. Kane (Eds.), *Improving Schools: Using What We Know* (pp. 212-252). Beverly Hills: Sage.

Gall, M.D., & Renchler, R.S. (1985). *Effective Staff Development for Teachers: A Research Based Model*. Eugene, OR: Clearinghouse on Educational Management, College of Education, University of Oregon. (ERIC/CEM Accession Number EA 017 615).

Huberman, M. & Crandall, D. *People, Policies, and Practices: Examining the Chain of School Improvement, Vol. IX*, Andover, Mass.: The Network, Inc.; 1982.

Joyce, B., & Showers, B. (1983). "Power in Staff Development Through Research on Training." Alexandria, VA: Association for Supervision and Curriculum Development.

Luehrmann, A. (1984, April). "The Best Way to Teach Computer Literacy." *Electronic Learning*, pp. 37-44.

Miles, M. B., Saxl, E. R., & Lieberman, A. (1985). "Key Skills of Educational 'Change Agents': An Empirical View." Prepared for *Curriculum Inquiry*, Center for Policy Research and Teachers College, New York, N.Y.

Moursund, D. (1985). *The Computer Coordinator*. Eugene, OR: International Council for Computers in Education.

AT FIRST YOU SUCCEED

Gail Sullivan

The Information Age

The Information Age is upon us. The Industrial Age has given way. The greatest indication of this change, we're told by economists, is that 60% of all American workers are involved in the creation, processing, and transmission of information.

While the economic world adjusts, the classroom teacher should feel right at home. Since the first 'Little Red Schoolhouse,' teachers have been collecting information, grading it, averaging it, sorting it, grouping it, reporting it to parents, and even affixing it to lifetime records.

Teacher Methods Over the Years

All teachers have dealt with these tasks routinely, and in a variety of ways. Intricate codes have been designed to reduce the amount of actual writing. Ingenious methods have been devised for recording scores, who got the bonus points on Spelling last week, who handed in their assignments late, as well as figuring out a way to note that Joe was absent PM only, dental appointment. Although some of these methods are passed on by tradition, most teachers have been on their own to develop systems.

A few years ago when accountability was getting attention, the school district began providing us with record books - a well-meaning gesture. But teachers spent hours searching for a pen small enough to record two digits in those teeny, tiny boxes. At the end of the ten week marking period, imposing twenty digit arrays filled the record book - hardly discernable as ten separate scores.

Teacher Workday was the signal for averaging the grades, and many brave souls spent hours at the calculator doing just that. The problem was that after this exhausting task, they didn't have the energy to evaluate the data.

It was T.S. Eliot who said, "Where is the knowledge lost in information?"

The Arrival of the Computer

Under the circumstances, deluged for decades with data requirements, the teacher should have been first in line when the computer arrived. Instead, even now in elementary schools, we find little general enthusiasm for taking advantage of this technological wonder. The computer is specifically designed to handle information and would relieve the teacher of tedious, routine tasks at the very least.

In the 1985 National Survey, Becker found that one-fourth of all United States teachers used computers "regularly" with students during the 1984-85 school year. A greater percentage of elementary teachers used instructional software with their students than secondary teachers; 37% versus 15%. When we examine the same groups of teachers for their use of word processors or other professional tools, the percentages are reversed. Only 10% of the elementary teachers used teacher utilities, while 30% of secondary teachers did so.

Altruism

Teachers have been altruistically learning about computers. They have done so in order to teach their students how to use computers. At the same time, teachers have been happy to have their students seated in front of the keyboard, rather than themselves.

Resistance

There has been a certain resistance to the machine by some teachers. "Emotionally, if not cognitively, we view computers as highly technical, somewhat mysterious machines that are used primarily by highly trained people, but not by us. That view of computers was nutured and encouraged by movies, television, and books for almost four decades in this society. Today, it is one of the major roadblocks to wider use and acceptance of computers in schools. We, as educators, are part of a transitional generation whose cultural training conflicts with current reality. We are uncomfortable with computers as machines we use directly, and our culture has encouraged and nurtured that view. Students, on the other hand, embrace computers and feel comfortable with them , and the social image of the computer encourages that (Willis,1984)." Contrast, for example, the foreboding presence of the computer HAL in the movie 2001: A Space Odyssey, with the computerized robot R2D2, of Star Wars. The movie 2001 presented HAL for one generation and they were impressed; today's students, on the other hand, encountered the robot R2D2 in Star Wars, and wished he lived down the block. The perceptions differed, and attitudes toward computers reflect the difference.

The attitudes are reflected at the computer itself. When I make a mistake at the computer, I stop and wonder what I did wrong. When my students make a mistake, they just push another key. The difference between reverence and love is operating here.

How can this problem be addressed?

Software to the Rescue

The hope for teachers is software - software that transforms the reverenced machine into a loveable helpmate.

It has begun to arrive. According to a survey by editors of Electronic Learning (May/June 1985), an encouraging trend has been noted in educational software. When they gathered the latest software from publishers in order to review it, the reviewers found they "unreservedly recommended more than half of the group (of 123 programs)." In 1984, they had recommended very few programs. It was a welcome indication that the quality of educational software was growing rapidly. During the recent school year (1985-86) software with improved content and curriculum application has been increasingly available. This trend was evidenced in instructional programs, not in utility programs. Utility programs have a long way to go to fullfill the needs of teachers. It is understandable since there has yet to be an infusion of capital and technology to increase productivity in the field of education (Moursund, 1986). Teachers have not been provided with adequate productivity aids.

While productivity software for teachers has not met the potential needs, software which is on the market can well serve as a beginning for teachers to take advantage of the computer's presence. Basic application packages have proved their value in the school setting over the past two years, and are a natural place to start.

An application program is a program for record keeping, word processing, authoring, or management. A teacher utility is an example of an application program.

Appleworks is such an application program gaining a prominent place in education. One review of the program stated that Appleworks "could substantially change the nature of education if its full potential were implemented by teacher, students, administration, and classified staff (Turner, 1985)."

Word Processors

Word Processing programs transform the computer into an electronically charged typewriter . . . and you have the gift of computerized writing. As you put your ideas into the computer, you can concentrate on the ideas, rather than worrying about the format, spelling or punctuation. With the power to rearrange whole blocks of text and to correct errors , the word processor is a powerful tool. It saves time. It allows the production of documents that have been fine-tuned. The ease of editing almost makes it fun to make changes. Add to that, the power to store your written efforts and call them up with ease to review or redo, and you can begin to grasp how this technology can serve needs.

Word processing is an easy way to introduce teachers to the idea of using the computer for reducing their work load. Beyond the student reports and parent letters, lie hidden powers. But the first step is to introduce word processing, and the first question the busy teacher asks when considering the idea is, "But how long will it take to learn to do word processing?"

Time Investment

Time is always an important consideration for busy teachers, so it is important to point out that "Yes, it is necessary to spend time to learn a word processing program." And then add quickly, "And yes, there is a payback for the investment."

The time investment for Appleworks (Word Processing only) is ten hours; one third of that time is for instruction, and two thirds is for practice. Or does ten minutes sound more like it? Richard Adams estimates it would take ten minutes for a new computer user to gain facility with Homeword by Sierra On-Line (Adams, 1985). With an icon driven menu, each function represented by an appropriate symbol, functions are quickly understood and easily accessed. Between ten hours and ten minutes are many word processors, and which one is chosen by educators often depends on availability.

The important point is that teachers begin to use computers, and word processing is guaranteed to be a gratifying beginning. A teacher plus a word processor is a great team that opens doors to further uses.

Printers

Teacher utilities cannot be serviceable without printers. Although similar in appearance to typewriters, printers need distinct handling. A base line knowledge of printers can be one way to reduce frustration for the new and experienced computer-using teacher. Little tips go a long way! How to tear the paper, change the ribbon, and set the pinfeed for labels are among the cursory bits of information that go far toward this goal.

Word Processing and knowledge of the printer could be just the beginning of a beautiful partnership between teachers and technology. The results of this partnership bring relief from tedious tasks and provide time for more important aspects of teaching. It adds professional touches to those letters and reports and proves satisfying to parents and students (Shadick and Stephenson, 1984). In other words, it rewards that time investment, and besides, it's fun! Learning an application program doubles as a "literacy course" for beginners, and surprisingly prepares the learner for other software. With careful choices, through recommendations from teachers and perusal of software reviews, the "new" computer-using teacher can find success in this computer environment.

During the presentation, these teacher utilities will be reviewed:
All of the Above, C & C Software.
Math Assistant I, Scholastic.
Spelling Worksheet Generator, Swift.
Printer tips will also be discussed.

References:

Adams, Richard C. "The Germ of an Idea - Infecting Others." The Computing Teacher, 19,9, (June,1985), p.51

Becker, Henry "Instructional Uses of School Computers." Reports from the 1985 National Survey, 1, (June 1986), pp. 1-10.

Editors "The Big Sift." Electronic Learning, (May/June, 1985), pp. 15-37.

Moursund, David "The Information Era: What Does It Mean To Education?" The Computing Teacher, 13, 5 (February, 1986) pp. 4-5.

Shadick and Stephenson "Computers in the Classroom, Helping Teachers Teach." The Computing Teacher, (August/September, 1984).

Turner, Jack The Computing Teacher, (October, 1984), pp. 34-35.

Willis, Jerry "Educational Computing: Current Status and Future Directions." Computers in the Schools, (Spring, 1984).

ROBOTS GO TO ELEMENTARY CLASSROOMS!

Elizabeth Anne Viau

Buzzing, beeping, red lights flashing, the little robot charges across the carpet. A kindergarten child crawls after it, intently watching the revolving cam in the trunk of the little vehicle. Are the robots here today? Do they have a place in elementary classrooms?

Inexpensive robots are indeed on the market. Eight of them were put in a kindergarten classroom to allow the children to experiment with robotic toys in the context of free play. Twenty children were enrolled in this class, which was made up of nine girls and eleven boys. The children were intensively observed as they interacted with the robots over a nine week period for a total of twenty-five hours and twenty minutes. In addition to the observations, the children were interviewed before and after the study. Some video-tapes were also made.

This paper will touch on the robots of our folklore and on what present day robots are really like. It will describe the robots that were used in the study, and explain what the children did with them. The paper will conclude with some suggestions for ways that robots might be used in other classrooms and at other grade levels.

Fantasy Robots

Say "robot" and what immediately springs to mind? For many of us, the answer is a kind of metal person, clumsy and clanking, perhaps, or vulnerable and talented, like C3PO in Star Wars. Western tradition can trace the dream of building intelligent, living creatures back into the mists of legend. Classical sculpture imitated the appearance of life; clock-work toys imitated its movement. Story tellers told of Pygmalian and Galatea, and whispered about the Golem.

With the advent of the industrial age, the dream took on more substance. Steam and gasoline engines replaced draft animals, drawing energy from new sources. If mechanical horses could be built, why not mechanical men? Science fiction stories, movies, and television, prepared the public for the appearance of metal people, producing technological fairy tales that included synthetic creatures who were very like, or even superior to, ourselves. But intelligent action requires intelligent direction, the capacity to receive feedback from the environment, and the judgement to make choices on the basis of situations as they are.

Electrical motors could run the limbs of sophisticated machines, but men were needed to direct them. Machines could not be programmed to perform a complex series of actions until the invention of the computer and the microchip. Now that work on programable systems is underway, industrial robots have appeared among us.

Real Robots

The industrial robots that actually exist today are very different from the robots of fiction. Many of them are only jointed arms with grippers. Programming them is difficult, but they are tireless and strong. The most primitive of them, the First Generation robots, are used for dangerous or unpleasant work in foundries or automobile factories; they excell at painting, welding, and lifting heavy loads.

Second Generation robots are also arms, although some of them are on rotating bases or tracks which allow them to move. These robots may have senses, such as cameras for vision. Their "hands" may consist of a group of interchangeable tools: screw-drivers, pneumatic suction pads, grippers and temperature sensors. They can do sensitive assembly work, and be reprogrammed for a variety of tasks.

Industrial robotics appears to be developing toward larger and larger systems, composed of a variety of collaborating machines under the control of linked computers. For example, a camera in the ceiling might communicate with a computer; the computer would direct a table to move under an arm so that work could be done. Eventually this line of development is expected to lead to large automatic factories, in which the internal parts are orchestrated in the machine production of products. Industrial robots can do many tasks, but they are not like the robots of our fantasies.

The robots that seem most promising for elementary education are the hobby robots. Some of these are toys, like the ones that will be described; some are more complex, commercially available robots for older students and hobbyists, and some are home-built robots that do things which amuse their owners. These robots are not very intelligent yet, but they are interesting, and fun to work with.

Robots for the Classroom

The robots that were used in the kindergarten were simple and inexpensive. They were all battery operated, and small enough for children to carry around easily. As some of the uses of robots remain to be discovered, the children were free to experiment with them. The study was concerned with what could be learned from observing the interaction of children with robots, rather than teaching any particular mode of using the toys.

Robots with "Visible Programs"

Three of the robots were wheeled toys from the Radio Shack. They were the Space Tank, the Space Robot and the Police Car. They cost about eight dollars each and can sometimes be purchased at half price on sale. They are made of red, white and blue plastic and each features a flashing light and sound effects. The Police Car is four wheeled, the Space Tank runs on treads, and the Space Robot has two powered wheels and a supporting caster. They are all directed by the same type of guidance system: a hexagonal post receives one of a set of six cams, and a follower traces the rim of the cam as the post rotates. The shape of the rim of the cam governs the path of the robot. These toys were chosen, in part, because the cams constitute "visible programs"; children could insert and remove the cams at will, and also watch the cams turning as they crawled after the robots. Because each robot has a different configuration of wheels, cams of any one particular shape would produce different movement patterns in each robot.

Robotic Arms

Two types of robotic arms were used. One was the Radio Shack Armatron, which costs thirty dollars. It is operated manually with two levers. The other, the Robotic Banker, is a small, twelve dollar "pick and place" robot, which picks up a penny and drops the coin in a slot. Then it moves a lever which spins a cylinder to reveal a word that supposedly answers the user's question. The Robotic Banker has an unvarying pattern of movements.

The Compurobot

Two Compurobots were also used. Toys-R-Us is a possible source for them; they are more expensive, costing thirty dollars each. However, a compurobot can do a lot. It has two wheels and two casters, and can be programmed by using a pad of command buttons on top of its "head". These buttons are rather like one-finger Logo: the operator presses a direction and a number for each command. The robot can remember a long string of commands and can go forward and back, and make turns. It also makes some noise as it operates.

The Petster

At forty dollars, the petster was the most expensive of the robots. It is a hemisphere covered with polyester fur, with a face and a stand-up tail. It has two motor-driven wheels. The petster responds to sounds, and is the only robot in the group that has any awareness of its environment. Once turned on, it responded to the children's claps, calls and stamping. The petster also responded to ambient noise in the classroom, turning and speeding off by itself from time to time. It, too, could go forward and back, turn right or left, meow, and purr.

Children and Robots

The children enjoyed the robots, and, despite the availability of many attractive alternate choices, continued to play with the robots enthusiastically throughout the study. The children were able to operate all the robots without difficulty, and even learned to program the Compurobot. Two features of their play were especially noticeable: the intensity of their physical interaction with the robots, and the strength of their curiosity about them.

Physical Play

Kindergarten children are doers. Given robots to play with, they played! They crawled after the robots as the toys traced patterns on the floor. They made bridges of their bodies and tried to get the robots to run under them. They put their arms down on the floor and tried to get the robots to run over their arms and hands. They crawled above the robots as the robots moved, keeping their bodies above the moving toys. They jumped back and forth over the robots as the robots ran. The children were very responsive to the movements of the robots, watching them closely, laughing, and responding quickly to any change in the direction of the robot's path.

Scientists in Action

Kindergarteners are keen observers. They watched the robots closely, and from every angle. In addition to crawling after the robots, watching the cams turn, they sometimes activated the robots and then turned the robots over, watching the wheels working. They tried many experiments, running robots over different materials, putting other toys on top of the robots as they ran, and running the robots so that they pushed against each other. A favorite game that the children invented was "fooling" the Robotic Banker; they would start the Banker's cycle by putting a penny in the slot, then snatch the penny away and laugh as the robot went through its routine. Sometimes they would put their own fingers in the Banker's gripper and go through the cycle instead of having a penny do so. The children tended to repeat their investigations, as if checking their results.

Robots and Schemata

Toys are often used as props for play. Doctor kits may stimulate games that let the children replay the schema of a visit to the doctor's office. What kinds of play would robots suggest?

The car and tank were used in conventional wheeled toy play, but the other robots did not come with ready-made scripts. The car and tank were also used in building schemes, which eventually led to the construction of block roads and ramps. Different types of play developed for other robots.

With the robotic arms, a favorite game was "Fool the Banker". Also observed were "It's Pinching Me" (the Armatron) and "Energy Exchange" where two boys hooked the two arms together so that the Armatron could "give energy" to the Robotic Banker. One boy played "Out of Control" with the Armatron, shaking and rattling the levers.

The girls sometimes played "Crazy Robots" with the Compurobots, programming them both and watching the two robots run around on the floor together. The boys played "Good Guy, Bad Guy" with the Compurobot and the Space Robot.

The Petster was the most popular robot, and some of the children treated it almost like a living pet. Individual children talked to the Petster, read to it, or engaged in animated conversation with it. One day several girls played at feeding the Petster plastic flowers, and another day one of the boys played at teaching the Petster to print a word.

Sex Differences

Overall, the boys played with the robots more than the girls. The favorite toy for the boys was the Space Tank; the girls loved the Petster. It is important, when choosing robots, to try to pick robots which will appeal to both girls and boys.

Things to Try

Even simple robots offer challenges. The kindergarteners enjoyed the opportunity to find out what the robots could do, and then began to use the robots in play with other toys. They were continuing to do so when the study ended, and were happily solving problems that developed naturally out of their play.

Older children, however, might enjoy more specific challenges. The three wheeled Radio Shack robots traced fairly large figures on the floor; how could the shapes of these figures be determined? Perhaps yarn on the floor, or markers on the turns, would provide information. Assigning the children to small groups facilitates discussion; each group could study a robot and then make predictions. For instance, where should each robot start, and with which cam, to pass through a "gate" between two blocks? The relationship between the shapes of the cams and the shapes of the robots' paths could lead to interesting mathematical speculations, especially as the answers are different for each robot.

The Armatron would add a new dimension to a science unit on the skeleton. Comparing natural and mechanical systems leads to a renewed respect for both. What sort of hand would a robot need to pick a flower? Pet a puppy? Children enjoy writing and illustrating; creating a book of special robot hands and their functions could spark a similar volume on the skills of human hands.

The Compurobot has commands similar to Logo commands. It can store a sequence of directives, and can be programmed to run through a maze. Some of its capabilities are quite sophisticated; it has a multiplication command, for instance, that will cause it to repeat a series of actions for a specified number of times. The Compurobot would be a good addition to units on measurement, angles, and mapping.

The Petster is controlled by the sounds that it hears, and a complex pattern of responses underlie its seemingly random movements. Figuring out the Petster's code involves theory building, and watching it run raises the question, "Is it alive?" What do we want of our pets, and can a robot satisfy our needs in this area? The Petster can respond to our signals, and would give a unit on communication an added dimension. Its appealing appearance make it a good subject for creative writing as well.

Robots are fun, and their potential has yet to be explored. The very thought of them has always been a spur to the imagination. Their makers are working on problems that nature has already solved, such as how to move from here to there, and how to get information about the environment. Nature's solutions become more interesting when we study the problems from a robot's perspective. More robots, and robots with sensors, are coming into the toy stores. Get ready! It won't be long until they start to come to school!

Suggestions for Further Reading

Chaille, C., & Littman, B. (1985) Computers in early education: The child as theory builder. In E. L. Klein (Ed.) Children and Computers. New Directions for Child Development, June 1985, pp. 19-35.

Forman, G. (1985b, June). The value of kinetic print in computer graphics for young children. In E. L, Klein (Ed.) Children and Computers. (pp. 19-35). New Directions for Child Development.

Gruber, H. E. (1973). Courage and cognitive growth in children and scientists. In M. Schwebel & J. Ralph (Eds.), Piaget in the Classroom (pp. 73-103). New York: Harper.

Kamii, C. (1973). Pedagogical principles derived from piaget's theory: Relevance for educational practice. In M. Schwebel & J. Ralph (Eds.), Piaget in the Classroom (pp. 132-154). New York: Harper.

Karmiloff-Smith, A. & B. Inhelder. (1974). If you want to get ahead, get a theory. Cognition, 1974, 3, 195-212.

Keller, J. & Shanahan. D. (1983, May). Robots in the kindergarten. The Computing Teacher, 10, 66-67.

Knight, D. C. (1983). Robotics: Past. Present and Future. New York: Morrow.

Larson, G. W. (1984, September). Robotics in elementary education. Robotics Age, 6, 21-22.

Marrs, T. (1985). The Personal Robot Book. Blue Ridge Summit: Tab.

Minsky, M. (Ed) (1985). Robotics. Garden City: Anchor.

Moore, M. (1985). Solving problems with robots". Conference on Computers and the Human Mind". Eugene: CATE.

Pritchard, S. F. (1984, March-April). Robots, bulldozers, and other map activities for the primary grades. Journal of Geography, 8, 1983-1984.

cnl
Suggested Books for the Classroom

Cummings, Richard. (1981). Make Your Own Robots. New York: McKay.

D'Ignazio, Fred. (1982). Working Robots. New York: Elsevier/Nelson.

Halacy, D. S. (1965). The Robots Are Here. New York: Norton.

Henson, Hilary. (1982). Robots. New York: Warwick.

Krasnoff, Barbara. (1982). Robots: Reel to Real. New York: Arco.

Litterick, Ian. (1984). Robots and Intelligent Machines. New York: Bookwright.

Reichardt, Jasia. (1978). Robots: Fact, Fiction, and Prediction. New York: Viking.

Ross, Dave.(1980). Making Robots. New York: Franklin Watts.

TEACHERS WRITE SOFTWARE WITH SUPERPILOT!

Elizabeth Anne Viau

Teachers can write software with music, text, graphics and animation, using the authoring language SuperPilot. Although SuperPilot takes some effort to learn, it is much easier to master than BASIC or Pascal. New users can produce finished instructional material, tutorials, tests, and drill-and-practice programs.

Why would teachers want to take the time to create software? Commercial software is getting better and more plentiful every month, and teachers have many demands on their energies. Yet there is a place in the classroom for software authored by teachers, a place that commercial software cannot fill.

During the winter quarter of 1986, this writer was the instructor of a class in Computer Assisted Learning at the University of Oregon. Twenty one teachers and instructional technologists took this class, and each person produced a tutorial written in SuperPilot. This paper will describe SuperPilot, discuss the process of software creation, and comment on the software that was produced.

An authoring language is a computer language that is designed for use in the development of interactive text programs. The commands are designed to be as simple as possible in order to make the language accessible to users who have no programming experience. Pilot is a popular authoring language, and versions are available for many brands of microcomputers. All the versions have similarities of syntax, so it is easy to transfer skills learned on one machine to the Pilot version that runs on another. SuperPilot is designed for use on the Apple IIe, and is a very rich and versatile version.

SuperPilot

SuperPilot comes with an authoring disk which is used in the creation of programs. The lessons that are created are saved onto blank disks, which can be duplicated and used in the classroom. Two disk drives are required for the production of lessons, but the lesson disks will boot in a single disk drive system if the lesson is called "hello". This arrangement protects the authoring disk and makes it possible for a teacher to make as many copies of the lesson as he desires.

The SuperPilot package also contains two Co-Pilot disks. These disks are tutorials and teach the use of the basic commands. Three volumes of documentation are also included; they are very comprehensive, but become most useful after the programmer has attained some facility with the language.

Lesson Text Editor

SuperPilot is subdivided into four editors. The most important of these is the Lesson Text Editor, which contains the text that will appear on the screen when the program is run. The Lesson Text Editor can ask questions, record answers, match answers to keys provided by the teacher, and branch to a remedial sequence if an answer is incorrect. The Lesson Text Editor also "calls" graphics, music and animation from the other editors.

Graphics Editor

The Graphics Editor allows the user to create and store pictures, which are drawn by using a segment of the keyboard as a graphics pad. Using the keys of the "pad", the programmer can direct a "turtle" on the screen to draw lines, squares, circles and irregular shapes. Lines of different colors can be drawn. Once the shapes are completed, they can be filled with color if desired. Pictures can be drawn directly on the screen, or can be traced under a clear plastic transparency with a graphic on it. A picture or map can be photocopied from a book, transferred to a thermal transparency, traced on to the screen, saved, and recalled to the screen on command.

There are two ways that the graphics can be recalled. In one mode, the graphics can be recalled line by line, as they were drawn. This slows down the program a little, but can give the effect of animation as the picture takes shape before your eyes. This process can be very effective, especially if a dynamic process is being represented, such as a volcanic eruption. The other mode causes the whole picture to appear on the screen in an instant.

Sound Effects Editor

The Sound Effects Editor allows the user to program beeps or musical scores. The notes of the melody, and their durations, are entered and saved as numbers and other typewritten symbols. These symbols can then be "called" by the Lesson Text Editor and played as the melody line of the song in the program as it runs. Sound effects can be used for emphasis in a lesson, to call attention to a specific point or frame, to set a theme, or to add an additional dimension to a character. It can spice up an introduction or set a tone for the lesson. One of the students, for instance, began a lesson about volcanoes with the theme from the song, "Ring of Fire".

Character Set Editor

The fourth editor is the Character Set Editor. This editor permits the design of alphabets with the lettering styles of your choice. Some alphabets come with the SuperPilot program, on the lesson program disk. This editor can also be used to create animation by drawing a picture on a grid and "calling" the parts that should appear on the screen through the Lesson Text Editor. In drawing an animal, for instance, the top part of the body could remain the same while differently configured legs are called under it as the body moves across the screen. The animal will appear to be running.

Lesson Design

Before the programming language can be used, however, the author must plan the lesson. The program may be a quick, simple quiz, a tutorial, or even a simulation.

Planning

The type of program and the topic of the lesson must be decided upon first. The next step is to choose the audience for the program: obviously, different grade levels will have different reading skills and background knowledge. Once the audience is chosen, the prerequisites for, and objectives of, the lesson can be chosen. In listing all these factors, the teacher-programmer clarifies what the lesson should include and exclude.

Excluding what the lesson will not be concerned with is difficult; as teachers, we really want to communicate, and in this phase of the planning the difference between human teachers and computer programs begins to be apparent. The computer does not see the puzzled look, and cannot judge when the student is just guessing. As teachers, we want to provide support for the students, to put in the little bit of background information that may have been forgotten, to clarify exactly what we mean with a few more examples. We do this automatically in our classrooms all day long; to do so in a computer program can easily result in a lifetime project! Objectives must be clear, coherent and simple; they must be direct as laser beam. This means that only a very simple topic can be chosen, e.g., how to clean a disk drive head, the meaning of one half and one fourth, or how to read a graduated cylinder. Even these limited topics generated complex programs. Good planning is critical to the success of the project.

Scripting

The second step is writing the script. The computer is a new medium, and the limited size of the screen imposes restraints on instruction. There is no easy leafing back, comparing past pages, as with a book: it is best to put all the information about a point on the screen at once.

Computer monitors are still a strain to read; to hold the learner, the material must be double-spaced, and attractively arranged with unbroken phrases. As the material is broken down into a screen-by-screen script, another demand is imposed by the medium; the strength of computer teaching is interaction with the individual, and so the designer must provide many opportunities for the learner to respond. Questions also take up space on the screen, and should give the learner an opportunity to practice responding correctly. These considerations result in a sequence of screens for each objective.

Storyboarding

Once the script has been written, the program is laid out on storyboards. Storyboards are used in planning many media productions, such as slide shows, movies, video-tapes and overhead projector presentations. For work on the computer, computer coding forms are useful; an area of forty spaces by twenty four spaces (the number of letters that can be printed on a single screen) is ruled off, and the lettering for each screen is laid out in the spaces. At this point, graphics are also roughly sketched in, so that the general appearance of each screen can be visualized. Experiments and changes can be made more easily on paper than on the screen.

Coding

The teacher is now ready to begin coding. SuperPilot has twenty two basic commands used in coding the lesson text. These commands are like verbs, e.g., T = type, M = match. There are also six conditioners which set the conditions under which the commands will be executed. For example:

```
*12              (this is screen 12)
G:es             (CLEAR screen)
T:  4 + 6 =      (TYPE "4 + 6 =" on screen)
A:               (ACCEPT input of answer)
M: 10!ten        (does answer MATCH 10 or ten?)
TY: Right!       (TYPE "Right!" on screen if the answer
                  given matches 10 or ten)
UY:flags         (USE (go to and run) subroutine
                  called flags if answer matches 10
                  or ten; return to this point)
JY:15            (JUMP to screen 15 if answer
                  matches 10 or ten)
TN: Try again!   (TYPE "Try again!" if answer did
                  not match 10 or ten)
R: (REMARK: screens 13 and 14 give more practice
                  on 6 + 4, then return to screen 12)
```

Using just a few commands, text is displayed, input is accepted, comparisons are made, subroutines are called, and, responding to input, the program can be made to branch. Each screen can be labelled individually by a name or a number, and accessed from any part of the program. It is also possible to count student responses, giving a score, or give different hints to the student on the first and second try at an answer. Records for each student can be saved on disk for later review by the teacher.

The teacher has considerable control over the matching process. By using symbols with the answers, such as "!" for or, she can specify whether the program should accept one answer, several answers, or answers with approximate spellings.

SuperPilot can be used by students as well as by teachers. Using the graphics editor and the TYPE command, children can create stories with pictures; adventure games can be written with the addition of ACCEPT, MATCH and JUMP. However, although SuperPilot can be used successfully at a simple level, it also has advanced capabilities that will not be soon outgrown.

Teacher Written Programs

But let us be realistic. Teachers are pressed for time! What kinds of software could teachers write, and why would they bother?

First, teacher-written software fits classroom instruction like a glove. The pieces written by the teachers were not designed to stand alone; they were seen as fulfilling some function in a lesson already being successfully taught. They echoed the teachers' own instructional strategies and served to provide support at some particularly difficult point. Some aimed at visual presentation of specific, basic concepts, such as latitude or fractions; for the visual learner, the absent child, or even the inattentive pupil, these programs provide opportunities for personal review.

Second, a collection of these programs would make good classroom "tool kits". Every subject matter area has basic facts and concepts, material that is introduced and taught once, yet intermittently in need of review with individuals. Examples of this type of information are how to use scientific equipment, concepts in measuring, or basic facts in biology or geography. Teacher written software, if available, would provide a way for students to review material that has been forgotten or is unclear to them.

Third, teacher-written software can serve special interests which do not support an economically viable market. Software manufacturers have bills to pay; they cannot afford to write programs of limited appeal. One student wrote an excellent program on bicycle gearing. This program would be a fine addition to a science unit on gears or the transfer of energy; however, the market for such a program may not be large enough to be profitable.

Fourth, once a teacher has learned to use SuperPilot, it is not difficult to write quick tests and review programs. A "Copy" function makes it possible to replicate a single frame as often as desired; the teacher has but to change the question and the acceptable answers in each frame. The computer can keep track of children's scores. Unlike a ditto, the program is always fresh, and can be used by children as their needs and wishes dictate. The program can be as simple or as complex as desired.

Fifth, writing an educational program makes the writer aware of the limitations of the computer as an instructional tool. The teacher can challenge the limitations, but she will also see the computer in a new light. She will find herself looking at software more knowledgeably. She will see software more critically as an educator, and more appreciatively as a programmer. She will feel more able to trust the computer because she knows what it does--and does not do--well.

Sixth, writing software with graphics and music makes the writer feel good--about herself, and about the computer. Creating that first program is hard work, but what a thrill it is to see it run! Because the writer has worked with the computer to produce something unique and creative, the computer is perceived as more accessible, more useful, and less mysterious.

Seventh, teachers share. Teacher written programs are sure to gradually circulate from classroom to classroom. They will be used, adapted, changed, improved. As modem use becomes routine, the programs will travel further and faster. Our class of twenty one students agreed to share their programs. Each student worked hard, each wrote one program, and each had the opportunity to leave the class with twenty one programs on a variety of topics. Individual investments of time pay off in a group of individuals who share.

In short, learning to write software in SuperPilot can help a teacher to integrate computers into her curriculum. It can lead to increased understanding of the computer as an instructional tool, and the creation of software to compliment her lesson plans and teaching style.

Suggestions for Further Reading

Burke, R. L., CAI Sourcebook. Englewood Cliffs: Prentice Hall 1982. 206 pages.

Meyer-Markle, Susan, Designs for Instructional Designers (2nd Ed.). Champaign: Stipes 1983. 221 pages.

Landa, R. K., Creating Courseware. New York: Harper & Row 1984. 295 pages.

CRITICAL ISSUES IN THE TEACHING OF WRITING WITH A WORD PROCESSOR

Keith Wetzel, University of Oregon

This study examines the extent to which a standard word processing program assists third, fourth, and fifth grade students' composing. In particular I am interested in whether the effects of word processing will depend on student typing skills. One possibility is that more efficient typists will derive a larger advantage from word processing because it eases the physical operations associated with transcribing. At the same time slower speed students may not be able to take full advantage of the word processor as their attention is devoted to lower level cognitive skills such as locating the correct keys. I hypothesized that students who were more efficient typists would score higher on measures of writing quality than students with less typing skill.

Additionally, I hypothesized that classes using the "Computers in Process Writing Program" would score higher on writing measures than classes using a traditional language arts program. Other researchers have found that regular elementary students who used computers along with a special writing program improved the quality of their writing (Levin, Boruta, and Vasconcellos, 1983; The NETWORK, 1984). In these investigations the control groups followed a traditional writing program and used pen or pencil as the writing tool. The study reported here is a modified replication of the above investigations.

Method

This ten week study used a pretest-posttest control group design. Eight classrooms of third, fourth, and fifth grade students were randomly assigned to two groups, one using computers in a process writing program; the other using their traditional language arts program and pen or pencil as the writing tool. Two weeks prior to the study, the computer-using group was instructed for two hours about keyboarding.

Each group of students produced two writing samples as a pretest to measure the quality of their writing. During the study the control group students followed their normal language arts program and used traditional tools of writing: paper and pencil. The experimental group students used the "Computer in Process Writing Program," a process approach toward writing instruction, aided by a computer and word processor. After these treatments each group produced another set of writing samples as a posttest measure of writing quality. All writing samples were scored holistically by trained raters. An analysis of covariance with pretest scores as the covariate was used to analyze the data.

Thirty-six students from the computer using group were randomly selected and tested for typing speed. These students were divided into slow typist and fast typist groups based on their typing speed. The writing scores of these students were analyzed using analysis of covariance with pretest scores as the covariate.

Results

Analysis of posttest findings indicated that the writing quality of the fast typists was significantly better than that of the slow typists ($p < .015$). Two explanations for the finding may be offered. First, researchers have indicated that faster writers are able to relieve some burdens on short term memory and this allows the writer to develop more fluency in writing. Daiute (1983) reports that faster typing means faster flow of ideas and that faster production of words on paper keeps memory free for other activities such as the development of content for the piece of writing. Second, the relation of the speed of writing to the quality of writing may be influenced by the procedure for the evaluation of writing samples through the holistic scoring by writing teachers. Writing teachers tend to rate long student papers higher (Daiute, 1985). If writing essay tests are timed, as they were in this study, students who are faster writers can write more in the allotted time, and are therefore more likely to receive higher scores.

The pre and posttest writing sample data were also analyzed to compare the "Computer in Process Writing Program" classes and the traditional language arts classes. The analysis indicated that the group using traditional language arts programs and pen or pencil as writing tool scored as well as the group using the "Computers in Process Writing Program" ($p < .93$).

This finding requires closer examination. It contrasts with the findings of Levin, Boruta, and Vasconcellos (1983), and The NETWORK (1984) which indicate that students who used a computer-based writing program did score significantly higher than control groups using a standard writing program and a pen or pencil as the writing tool. However, in these studies, researchers required all students to complete writing samples using pen or pencil rather than computer. Students following a computer-based writing program used pen or pencil for the writing test samples as did the students following a traditional language arts curriculum. In contrast, students using the "Computers in Process Writing Program" in my study wrote pretest and posttest writing samples using the computer. This change in procedure allowed students who practiced writing with computers to use the computer to complete pre and post writing samples.

Contrary to my expectation, this change in procedure may have placed the computer using group at a disadvantage because this group typed more slowly than they wrote by pen or pencil. The copying speed of most students in the intermediate elementary grades by pen or pencil is 8.8-12.5 words per minute (Groff, 1961). I found that the average copying speed by pen or pencil of students in one fourth grade classroom in the study was 8.9 words per minute. In contrast the average copying speed by keyboard in this study was five words per minute. Thus, the physical capacity of "Computer in Process Writing Program" students to generate text was less than their counterparts in the control group who used pen or pencil. As explained above the speed of writing influences both writing fluency and the length of writing samples. Consequently, slower keyboard entry may have depressed the writing scores of computer-using-students.

The change in testing procedure may account for the difference is results between this study and those of Levin, Boruta, and Vasconcellos (1983) and The NETWORK (1984).

Conclusions

On the basis of the study findings, the following conclusions can be drawn:

1. Students using the keyboard were less comfortable than those using pen or pencil. Students may require more training in word processing and keyboarding before they are able to achieve as much as they were able to achieve with paper and pencil.

2. There is evidence that students who have better typing skills do improve their writing to a greater degree.

3. Elementary students should have keyboarding instruction before they use the computer for writing. Keyboarding instruction should continue until students type as fast as they write with pen.

In addition to the above conclusions, there were several observations reported by student and staff: (a) Teachers reported that students in the "Computer in Process Writing Program" wrote more and received more feedback on their writing than they had previously. (b) Students enjoyed the computers and writing program; they preferred the computer to paper an pencil. (c) Finally, the principal and the teachers reported that parent response to this program was clearly favorable.

REFERENCES

Daiute, C. (1983). The computer as stylus and audience. College Composition and Communication, 34, 134-145.

Daiute, C. (1985). Writing and computers. Reading, MA: Addision-Wesley Publishing Company, Inc.

Groff, P. (1961). New speeds of handwriting. Elementary English, 38, 564-565.

Levin, J. A., Moruta, M., & Vasconcellos, M. T. (1983). Microcomputer-based environments for writing: a writer's assistant. In A.C. Wilkinson (Ed.), Classroom Computers and Cognitive Science (pp. 219-232). New York: Academic Press.

NETWORK (1983-84). Quill JDRP application. 290 South Main Street, Andover, MA 01810.